TESIS - IF185401

PENGEMBANGAN MEKANISME *CLUSTER HEAD ADOPTION* UNTUK EFISIENSI ENERGI *LEACH-BASED CLUSTERING PROTOCOL* PADA *WIRELESS SENSOR NETWORK*

RYAN RIZKI ADHISA
NRP. 5116201061

DOSEN PEMBIMBING

Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
NIP. 197410222000031001

PROGRAM MAGISTER

BIDANG KEAHLIAN KOMPUTASI BERBASIS JARINGAN

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2019

[Halaman ini sengaja dikosongkan]

TESIS - IF185401

THE DEVELOPMENT OF CLUSTER HEAD ADOPTION MECHANISM FOR ENERGY EFFICIENT LEACH-BASED CLUSTERING PROTOCOL IN WIRELESS SENSOR NETWORK

RYAN RIZKI ADHISA
NRP. 5116201061

SUPERVISOR

Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
NIP. 197410222000031001

MASTER PROGRAM

DEPARTMENT OF INFORMATICS

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2019

[Halaman ini sengaja dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Ryan Rizki Adhisa
NRP.5116201061

Dengan judul :
Pengembangan Mekanisme *Cluster Head Adoption* Untuk Efisiensi Energi *LEACH-Based Clustering Protocol* Pada *Wireless Sensor Network*

Tanggal Ujian : 10 Januari 2019
Periode Wisuda : Maret 2019

Disetujui oleh :

Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
NIP. 197410222000031001 ..... (Pembimbing 1)

Royyana Muslim I, S.Kom., M.Kom., Ph.D.
NIP : 197708242006041001 ..... (Penguji 1)

Tohari Ahmad, S.Kom., MIT, Ph.D.
NIP : 197505252003121002 ..... (Penguji 2)

Dr.Eng. Radityo Anggoro S.Kom., M.Sc.
NIP. 198410162008121002 ..... (Penguji 3)

Dekan Fakultas Teknologi Informasi,
dan Komunikasi

Dr. H. Agus Zainal Arifin, S.Kom., M.Kom.
NIP. 197208091995121001

[Halaman ini sengaja dikosongkan]

PERNYATAAN KEASLIAN

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tesis saya dengan judul:

PENGEMBANGAN MEKANISME *CLUSTER HEAD ADOPTION* UNTUK EFISIENSI ENERGI *LEACH-BASED CLUSTERING PROTOCOL* PADA *WIRELESS SENSOR NETWORK*

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pusaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 15 Januari 2019

Ryan Rizki Adhisa
NRP: 5116201061

[Halaman ini sengaja dikosongkan]

Pengembangan Mekanisme *Cluster Head Adoption* Untuk Efisiensi Energi *LEACH-Based Clustering Protocol* Pada *Wireless Sensor Network*

Nama Mahasiswa : Ryan Rizki Adhisa
NRP : 5116201061
Pembimbing I : Waskitho Wibisono, S.Kom, M.Eng., Ph.D.

ABSTRAK

Wireless Sensor Network (WSN) adalah sebuah teknologi yang memungkinkan untuk melakukan pengamatan terhadap kejadian / fenomena yang terjadi pada cakupan wilayah yang sangat luas dengan menggunakan perangkat komputer berukuran ringkas *(sensor node)* dan metode komunikasi secara nirkabel. Dalam WSN digunakan banyak *sensor node* yang fungsinya untuk mengumpulkan informasi seperti fenomena perubahan suhu, intensitas cahaya, panas, dll., dengan sebuah / beberapa pusat pengumpulan data *(sink)* dengan cara berkomunikasi melalui gelombang radio jarak dekat. *Node* pada WSN memiliki energi, kecepatan komputasi dan juga kapasitas memori yang terbatas sehingga perlu digunakan algoritma routing yang dapat meningkatkan performa jaringan.

Low-Energy Adaptive Clustering Hierarchy (LEACH) adalah salah satu algoritma *routing* WSN berbasis *cluster*. Ide utama LEACH adalah untuk membentuk *cluster* dengan sebuah *node* sebagai *Cluster Head (CH)* yang fungsinya untuk mengumpulkan data dari *member node* agar selanjutnya komunikasi dengan *sink* dapat dilakukan oleh CH tersebut saja. Pada LEACH pergantian CH dilakukan berdasarkan waktu, dimana sebuah sesi disebut *round*. Setiap *round* baru maka tiap *node* akan melakukan perhitungan berdasarkan sebuah rumus probabilitas untuk menentukan apakah dirinya menjadi CH apa tidak sehingga persebaran CH tiap *round* dapat dikatakan acak. Hal ini dapat menimbulkan masalah *multiple CH*, dimana terbentuk banyak CH yang berdekatan sehingga terlalu banyak CH *advertisement* yang dikirim. Selain itu dikarenakan jangkauan radio dari *node* terbatas, maka dapat terjadi masalah *stray node* dimana sebuah *node* yang tidak menjadi CH tidak menerima CH *advertisement*. Oleh sebab itu pada penelitian ini diusulkan sebuah metode pemilihan CH dalam masalah *multiple* CH yang *energy-aware* dimana CH dengan tingkat *residual energy* yang tertinggi akan diadopsi oleh *member node*. Untuk menyelesaikan masalah *stray node*, dilakukan pembentukan *cluster* dengan cara menjalankan ulang fase *setup* LEACH oleh *stray node* yang memiliki tetangga beberapa *stray node* lain yang tingkat energinya lebih rendah. Dari Hasil penelitian menunjukkan, bahwa dengan mengimplementasikan kedua metode yang diusulkan dapat meningkatkan *network lifetime* hingga 779 menit, PDR meningkat 3%, dan juga menghasilkan *residual energy* rata-rata pada T-1000, T-2000, dan T-3000 menit masing-masing lebih tinggi 1,7%, 2,4%, dan 1,9%.

Kata kunci : *LEACH, Multiple CH, Stray Node, WSN*

[Halaman ini sengaja dikosongkan]

The Development of Cluster Head Adoption Mechanism for Energy Efficient LEACH-Based Clustering Protocol in Wireless Sensor Network

Name of Student : Ryan Rizki Adhisa
NRP : 5116201061
Supervisor I : Waskitho Wibisono, S.Kom, M.Eng., Ph.D.

ABSTRACT

Wireless Sensor Network (WSN) is a technology that enabled us to monitor natural phenomena in a vast area using wirelessly connected small form-factor computer hardware called sensor node. In WSN many sensor nodes are used to collect information such as changes in temperature, light intensity, etc., with a single data collector used called sink through a short-radio wave communication. Typically, nodes in WSN have limited energy, computational speed, and memory capacity so an efficient routing algorithm is needed to maximize the network performance.

Low-Energy Adaptive Clustering Hierarchy (LEACH) is one of the cluster-based routing algorithms. The main idea of LEACH is to form a cluster with a single node that acts as cluster head (CH) to collect sensed data from member node so then the data is sent to the sink by only the CH to conserve energy. In LEACH CH job is rotated based on a time called round. At the beginning of each new round, every node in the network will do a probabilistic calculation to determine whether it becomes a CH or not for that round so the CH is randomly formed. This can cause multiple CH situations, where more than one CH is formed closely together and the same member node is receiving more than one CH advertisement message. Other than that because of limited radio coverage of CH, some of the nodes in the network will not receive any of the CH advertisement message thus become a stray node. In this research, a method to deal with the problems mentioned is proposed. For the multiple CH situations, CH with higher residual energy will be adopted by member node. For the stray node problem, a second cluster forming mechanism with energy consideration is proposed so that stray nodes can form its own clusters. The results of the experiment show that with the proposed method implemented, network lifetime can be increased by 779 minutes, Packet Delivery Ratio (PDR) can be improved by up to 3%, and the average residual energy of the network at T-1000, T-2000, and T-3000 minutes is higher by 1,7%, 2,4%, and 1,9% respectively.

Keywords: LEACH, Multiple CH, Stray Node, WSN

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji syukur penulis panjatkan kepada ALLAH.SWT atas limpahan rahmat dan petunjuk-Nya, sehingga penulis mampu menyelesaikan tesis yang berjudul "Pengembangan Mekanisme *Cluster Head Adoption* Untuk Efisiensi Energi *LEACH-Based Clustering Protocol* Pada *Wireless Sensor Network*".

Dalam proses pembuatan tesis ini sangat banyak bantuan – bantuan yang penulis terima dari berbagai pihak, untuk itu penulis ingin mengucapkan terima kasih sebesar-besarnya kepada :

1. Ibu dan Ayah penulis, Dra. Hartani Inggarsasi dan Ir. Joni Maryono M.T..
2. Kekasihku Lendy Ivoni Bramanda, S.T..
3. Kakak kandungku Sandy Rezadati S.I.Kom.. dan istrinya Karina Astrid Anastasya S.E.
4. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku pembimbing dalam menyelesaikan tesis ini yang sangat banyak membantu penulis dengan memberikan kepercayaan, nasehat, solusi dan pembelajaran serta dukungan sehingga penulis dapat menyelesaikan tesis ini. Semoga Allah SWT membalas semua jasa baik bapak.
5. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D., Bapak Tohari Ahmad S.Kom., MIT., Ph.D. dan Bapak Dr.Eng. Radityo Anggoro S.Kom., M.Sc., selaku dosen penguji tesis ini dan juga telah banyak memberikan saran untuk penyempurnaan tesis ini.
6. Bapak dan Ibu Dosen Prodi Magister Teknik Informatika ITS yang telah memberi bimbingan perkuliahan sehingga menambah wawasan dan pengetahuan penulis.
7. Kementrian Pendidikan dan Kebudayaan (Kemdikbud) yang telah menganugerahi Beasiswa Unggulan kepada penulis sehingga penulis dapat menyelesaikan perkuliah Magister Teknik Informatika di Institut Teknologi Sepuluh Nopember.
8. Kak Rozita dan Vynska Amalia Permadi serta seluruh teman-teman angkatan 2016 yang telah membantu dan memberi dukungan kepada penulis.
9. Romio si kucing peliharaanku yang telah banyak mengganggu dalam proses penulian tesis ini.

Tesis ini masih jauh dari kesempurnaan. Besar harapan penulis untuk dapat menerima kritik dan saran, sehingga dapat menjadi bahan perbaikan di penulisan berikutnya.

Surabaya, 15 Desember 2018

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Dengan perkembangan Teknologi Informasi (TI) terutama di bidang mikroprosesor, kegiatan manusia dapat ditunjang dalam berbagai hal seperti pada bidang kesehatan, pengawasan lingkungan hingga militer. Hingga saat ini banyak bermunculan mikroprosesor yang semakin efisien dan kecil ukurannya sehingga dapat dimanfaatkan untuk membuat sebuah perangkat komputer yang ringkas dengan berbagai macam fungsi. *Wireless Sensor Network (WSN)* adalah sebuah teknologi yang memungkinkan untuk melakukan pengamatan terhadap kejadian / fenomena yang terjadi pada cakupan wilayah yang sangat luas dengan menggunakan perangkat komputer berukuran ringkas *(sensor node)* dan metode komunikasi secara nirkabel (*wireless)* (Jino Ramson & Jackuline Moni, 2017). Dalam WSN digunakan banyak *node* yang fungsinya untuk mengumpulkan informasi seperti fenomena perubahan suhu, intensitas cahaya, panas, dll., dengan sebuah / beberapa pusat pengumpulan data *(sink).* Informasi yang berasal dari *sensor node* dikirimkan kepada *sink* dalam bentuk *response query* dengan menggunakan protokol *routing* tertentu. Informasi yang sampai pada *sink* kemudian dapat diolah atau disimpan pada pusat data.

Sensor node pada WSN berkomunikasi melalui gelombang radio jarak dekat baik itu untuk keperluan pengiriman data ataupun koordinasi dengan *sensor node* disekitarnya untuk keperluan *routing.* Pada WSN umumnya digunakan *node* yang memiliki keterbatasan dalam hal energi, kecepatan komputasi dan juga kapasitas memori sehingga perlu digunakan algoritma routing yang dapat memaksimalkan *Network Lifetime* dan *Packet Delivery Ratio.* Sudah banyak penelitian yang dilakukan untuk meningkatkan efisiensi algoritma *routing* pada WSN dengan salah satu contoh algoritma *routing* yang paling banyak dibahas adalah *Low-Energy Adaptive Clustering Hierarchy (LEACH)* (Raed & Abdalraheem, 2013) .

LEACH adalah salah satu algoritma *routing* WSN berbasis kluster yang paling banyak dibahas pada penelitian. Ide utama LEACH adalah untuk memilih sebuah *node* sebagai *Cluster Head (CH)* yang fungsinya untuk mengumpulkan data dari *node* membernya agar selanjutnya komunikasi dengan *sink* dapat dilakukan oleh CH tersebut saja (Liu, 2012).

Pada LEACH pergantian CH dilakukan berdasarkan waktu, dimana sebuah sesi waktu disebut *round*. Pada saat awal *round* baru dimulai maka tiap *node* pada jaringan akan memilih sebuah angka acak X dimana 1<X<0 dan membandingkannya dengan nilai *threshold* yang digunakan pada suatu *round* yang dihitung dari sebuah fungsi probabilitas yang digunakan dalam LEACH untuk menentukan apakah dirinya menjadi CH apa tidak pada *round* tersebut. Hal ini bertujuan untuk merotasi jabatan CH secara merata agar tidak ada *node* yang lebih cepat mati karena dengan menjadi CH konsumsi energi sebuah *node* akan lebih tinggi (Palan, Barbadekar, & Patil, 2017).

Masalah yang timbul dari hal tersebut adalah persebaran CH yang cenderung acak untuk tiap *round* dan memungkinkan untuk terbentuk banyak CH yang berdekatan pada suatu *round* (*multiple CH). Member node* yang berada pada jangkaun radio beberapa CH akan menerima pesan CH *advertisement message* dari semua CH tersebut sehingga perlu diaplikasikan sebuah solusi pemilihan CH yang efektif agar performa jaringan meningkat.

Selain itu dengan jangkauan radio CH yang terbatas maka memungkinkan untuk beberapa *node* tidak mendapat pesan CH *advertisement* sehingga tidak terhubung pada CH manapun sehingga *node* tersebut (disebut *stray node* dalam penelitian ini) tidak tergabung pada *cluster-cluster* yang terbentuk dan tidak dapat mengirimkan data hasil *sensing* yang dilakukannya.

Telah diusulkan sebuah metode penanganan keadaan *multiple* CH dan *stray node* pada *Multi-hop* LEACH (M-LEACH) (Wibisono, Ahmad, & Anggoro, 2018). Pada penelitian tersebut digunakan parameter *distance* untuk memilih CH pada masalah *multiple* CH, dimana CH dengan jarak terdekat dengan *sink* akan dipilih oleh *member node*. Untuk masalah *stray node* pada penelitian tersebut digunakan model komunikasi *multi-hop* dengan *Shortest Geopath Routing* (SGR) / *Distance-based Greedy Perimeter Stateless Routing* (GPSR) yang prinsipnya

adalah memilih *node* yang memiliki jarak lebih dekat dengan *sink* sebagai alamat *next-hop* untuk meneruskan pesan dengan tujuan untuk mencari CH. Pada penelitian tersebut faktor energi tidak dipertimbangkan dalam penanganan masalah *multiple* CH maupun *stray node*, padahal dalam beberapa algoritma *routing* WSN yang telah ada jika faktor energi dipertimbangkan maka akan dapat menambah *network lifetime* (Liu, 2012).

Oleh sebab itu pada penelitian ini dikemukakan sebuah metode pemilihan CH pada masalah *multiple* CH yang *energy-aware* dengan tujuan agar *member node* dapat memilih CH terbaik dari sisi *residual energy*. Selain itu diusulkan sebuah metode penanganan *stray node* dengan cara membentuk *cluster* dari sekumpulan *stray node* dengan mempertimbangkan faktor energi dari para *stray node*.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini adalah sebagai berikut :

1. Bagaimana masalah *multiple* CH dapat ditangani secara *energy-aware*?
2. Bagaimana masalah *stray node* dapat diselesaikan dengan mempertimbangkan faktor energi?
3. Bagaimana pengaruh metode penanganan masalah *stray node* dan *multiple* CH yang diusulkan terhadap *packet delivery ratio*, *network lifetime* dan *latency*?

1.3. Batasan Masalah

Batasan masalah pada penilitian ini adalah :

1. Simulator yang digunakan adalah simulator WSN SIDNet-SWANS.
2. Mobilitas *node* yang disimulasikan adalah statis.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah untuk meningkatkan kinerja LEACH-*based clustering protocol* dengan menerapkan metode pemilihan CH yang *energy-aware* untuk masalah *multiple CH* dan metode agar *stray node*

dapat mengirim data hasil *sensing* yang dilakukannya menuju *sink* dengan cara membentuk *cluster* dengan mempertimbangkan faktor energi.

1.5. Manfaat Penelitian

Manfaat penelitian ini adalah mengembangkan metode LEACH-*based clustering protocol* dalam hal penanganan masalah *multiple* CH dan *stray node* yang timbul jika digunakan parameter transmisi radio sesuai *hardware* WSN nyata.

1.6. Kontribusi

Kontribusi dari penelitian ini adalah sebagai berikut:

1. Diusulkan metode pemilihan CH pada masalah *multiple* CH yang *energy-aware* dengan mempertimbangkan *residual energy* dari CH *node*.
2. Diusulkan metode penanganan *Stray Node* yang timbul pada LEACH dengan cara melakukan fase *setup* LEACH kedua dengan pertimbangan *residual energy* pada *node* yang tidak mendapatkan CH pada suatu *round* dan memiliki tetangga *stray node* lainnya yang energinya lebih rendah.

BAB 2
KAJIAN PUSTAKA

2.1. *Wireless sensor network*

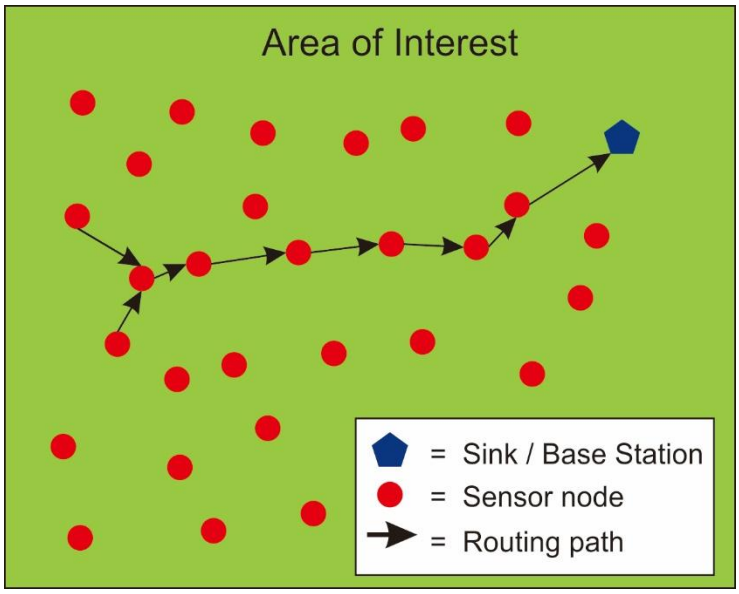
Wireless Sensor Network (WSN) adalah sebuah model jaringan komputer yang terdiri dari banyak perangkat yang terdistribusi pada suatu area tertentu dengan koneksi nirkabel dengan suatu tujuan tertentu. Pada komunikasi *wireless* dikenal 2 istilah yaitu *transmitter* dan *receiver*. *Transmitter* pada komunikasi nirkabel berfungsi sebagai pengirim pesan informasi sedangkan *receiver* berfungsi sebagai penerima dari pesan informasi. *Transmitter* dan *receiver* pada perangkat wireless umumnya sudah terintegrasi menjadi satu perangkat. Sehingga perangkat tersebut dapat bertindak sebagai pengirim atau penerima pesan informasi. Selain hubungan *transmitter* dan *receiver*, perangkat komunikasi nirkabel dapat dibedakan menjadi dua bagian yaitu *infrastructure devices* dan *field devices*.

Infrastructure devices merupakan perangkat nirkabel yang umumnya bertindak sebagai kordinator dari komunikasi tersebut dan memiliki kemampuan resource yang lebih tinggi dari pada *field devices*, pada WSN dikenal sebagai *sink*. Sedangkan *field devices* umumnya bertindak sebagai pekerja dari *infrastructure devices* yang bertugas mengumpulkan data dari lingkungan sekitar dan mengirimkannya kepada *infrastructure devices*, pada WSN dikenal dengan sebutan *sensor node*.

Sensor node pada WSN bertugas untuk mengumpulkan data dari suatu fenomena yang perlu diamati seperti suhu, kelembaban udara, tingkat radiasi, dan lain-lain. Tiap *node* pada WSN memiliki perangkat komunikasi nirkabel yang berfungsi untuk bertukar informasi dan data dengan tujuan umum adalah agar data yang didapatkan dari proses *sensing* dapat dikirimkan kepada pusat pengumpul data (*central station* / *sink)* (Jino Ramson & Jackuline Moni, 2017).

Dikarenakan keterbatasan kemampuan node penyusun WSN, salah satunya dalam jangkauan transmisi modul komunikasi nirkabelnya, maka umumnya dilakukan *message passing* diantara node penyusun WSN agar data dapat sampai pada *sink* terlihat seperti pada Gambar 2.1.

Agar efisien dalam proses pengiriman data, WSN memerlukan protokol *routing* yang berfungsi mengontrol bagaimana data dikirimkan dengan berbagai pertimbangan seperti jarak *node* dengan *sink*, jumlah *neighbour*, faktor energi, dan sebagainya.



Gambar 2.1. Contoh Alur Pengiriman Data pada WSN.

2.2. Clustering pada WSN

Teknik *clustering* pada WSN umumnya digunakan dengan tujuan penghematan sumber daya energi. Secara umum *clustering* pada WSN dapat digolongkan menjadi 2 kriteria berdasarkan cara membangun *cluster* yaitu *bottom-up* dan *top-down*. Pada pendekatan *top-down* akan ditentukan area *cluster* terlebih dahulu untuk kemudian dipilih siapa *cluster head node* dan *member node* untuk *cluster* yang telah dibentuk. Kekurangan dari metode ini adalah harus diterapkan model kontrol jaringan yang *centralized* sehingga rentan terhadap masalah komunikasi pada jaringan dan bentuk *cluster* yang terbentuk akan cenderung tidak fleksibel.

Pendekatan yang lain adalah *bottom-up* dimana *node* penyusun jaringan akan membentuk *cluster* dengan cara memilih *cluster head*. Pada model jaringan ini tidak ada sistem kontrol terpusat dan proses pembentukan *cluster* benar-benar

decentralized sehingga lebih tahan terhadap gangguan komunikasi pada jaringan. Selain itu bentuk *cluster* yang tercipta akan fleksibel sesuai dengan keadaan yang dipertimbangkan pada saat proses pembentukan *cluster*. Salah satu algoritma *routing* yang menerapkan model ini adalah *Low Energy Adaptive Clustering Hierarchy* (LEACH).

2.2.1. LEACH

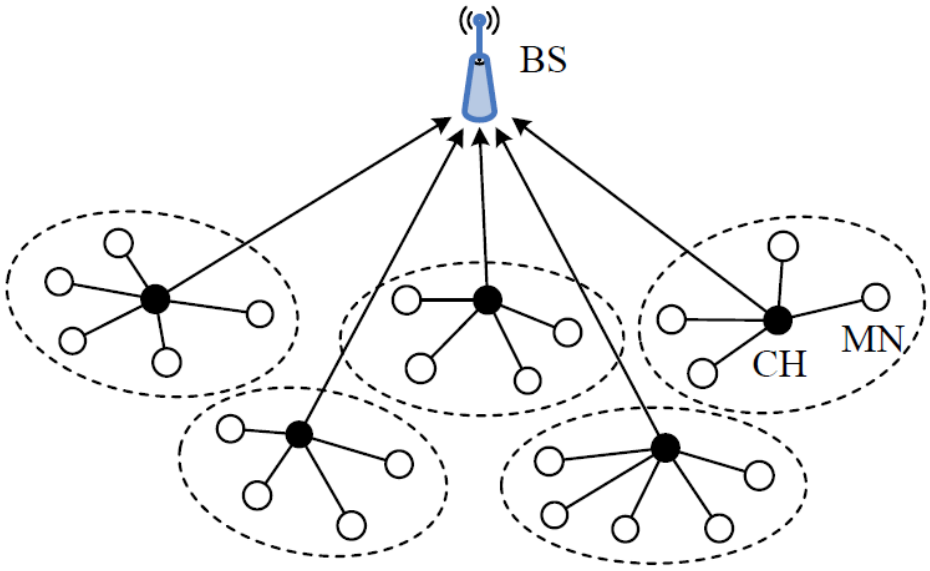
Low Energy Adaptive Clustering Hierarchy (LEACH) adalah sebuah *routing protocol* berbasis *cluster* dengan pendekatan *bottom-up*. LEACH membentuk *cluster* dengan sebuah *cluster head (CH)* dengan beberapa *member node* yang bergabung berdasarkan tingkat kekuatan transmisi radio yang terbesar untuk menentukan CH mana yang akan dipilihnya untuk meneruskan informasi dari proses *sensing* menuju *sink* (Raed & Abdalraheem, 2013), (Heinzelman, Chandrakasan, & Balakrishnan, 2000).

Pada LEACH ada dua fase yang berlangsung berupa fase *setup* dan fase *steady state*. Fase *setup* dimulai dengan pemilihan suatu *node* sebagai c*luster head* (CH) yang ditentukan secara acak berdasarkan level energi yang tersisa pada interval waktu tertentu. Pada interval berikutnya, setiap *node* sensor n mengambil sebuah bilangan acak x sedemikian rupa sehingga $0<x<1$, dan membandingkannya dengan *threshold* tertentu T(n) yang digunakan pada suatu *round*. Jika $x < T(n)$, maka *node* tersebut menjadi *cluster head* pada *round* tersebut, dan jika tidak maka akan menjadi *member node*.

Selanjutnya CH akan mengirimkan pesan *broadcast* ke *node* sensor dan jika *node* sensor menerima pesan tersebut akan membalas dengan pesan bahwa *node* sensor tersebut akan bergabung sebagai anggota c*luster*. Jika node yang bukan CH menerima pesan *broadcast* dari lebih dari 1 CH maka akan ditentukan dengan tingkat kekuatan transmisi tertinggi (Raed & Abdalraheem, 2013). Dikarenakan LEACH membentuk *cluster* dengan cara menentukan CH terlebih dahulu dan kemudian *node* lainnya bergabung sebagai *member* dengan bentuk cluster yang berbeda tiap *round* maka LEACH dapat dikatakan membentuk *cluster secara bottom-up* (Abushiba, Johnson, Alharthi, & Wright, 2017).

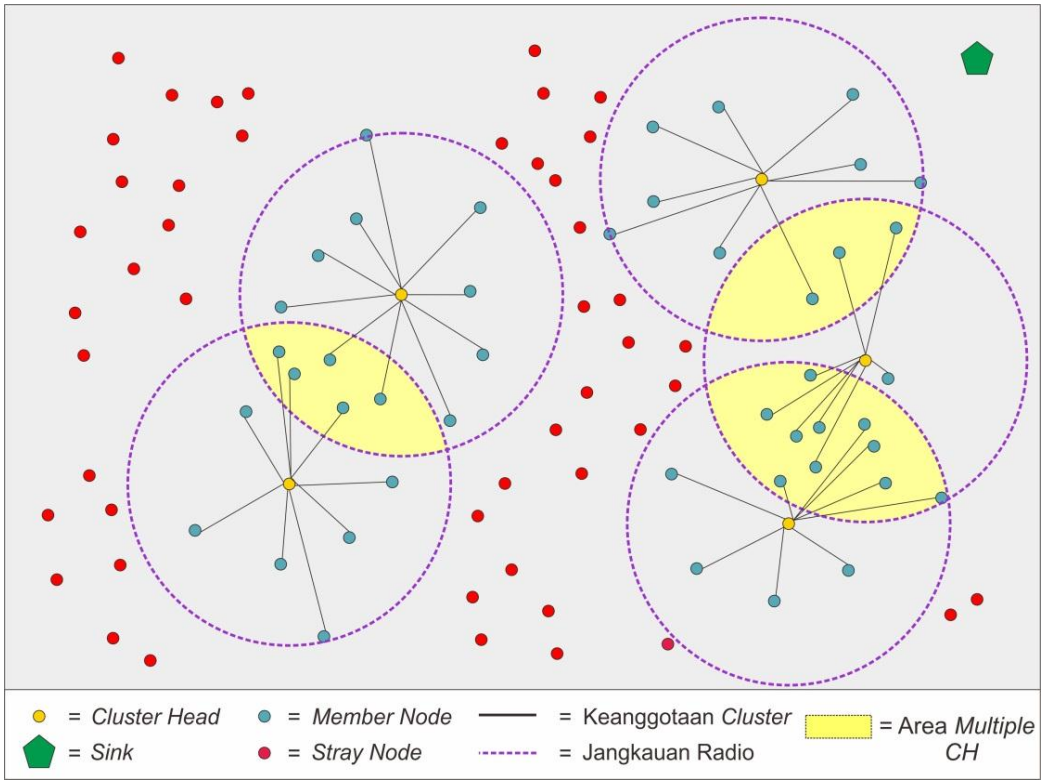
Pada fase *steady state* dilakukan pengiriman data hasil *sensing* dari *member node* menuju CH dan juga dari CH menuju *sink / base station (BS)*.

LEACH dapat menghemat energi karena hanya CH yang melakukan pengiriman data ke *sink* sehingga konsumsi energi *member node* berkurang dan menyebabkan *network lifetime* jadi lebih tinggi. Topologi dasar LEACH secara sederhana dapat direpresentasikan pada Gambar 2.2.



Gambar 2.2. Topologi dasar LEACH (Liu, 2012)

Dikarenakan tiap *round* baru tiap *node* akan menjalankan perhitungan probabilitas untuk menentukan dirinya menjadi CH apa tidak, maka bentuk *cluster* yang terbentuk akan cenderung berbeda pada tiap *round* pada LEACH. Hal ini memungkinkan juga menimbulkan *multiple CH* dan *stray node*, seperti yang ditunjukkan pada Gambar 2.3.



Gambar 2.3. Contoh *Multiple* CH & *Stray Node* Yang Terbentuk Pada Fase *Setup* LEACH.

2.2.2. *Stray Node* Pada LEACH

Pada WSN umumnya jangkauan radio dari suatu *node* terbatas, oleh sebab itu sangat mungkin pada sebuah algoritma *routing* berbasis *cluster* untuk timbul masalah *stray node*, yaitu kondisi dimana suatu *node* pada jaringan tidak tergabung dengan *cluster* yang terbentuk. Pada LEACH *stray node* terjadi jika pada saat fase *setup node* yang tidak menjadi CH tidak mendapatkan pesan CH *advertisement*.

Dalam algoritma asli LEACH, masalah ini tidak muncul, sebab diasumsikan bahwa semua *node* pada jaringan dapat berkomunikasi satu sama lain secara *one-hop* (Raed & Abdalraheem, 2013). Oleh sebab itu asumsi algoritma asli LEACH ini kurang cocok jika diaplikasikan pada WSN skala besar, karena dapat menyebabkan masalah *overhearing* (Maimour, 2018), selain itu secara fisik sebuah *hardware sensor node* penyusun WSN tidak akan dapat menjangkau *node*

yang lain pada jarak yang jauh dikarenakan keterbatasan jangkauan radio (Karray, 2018).

Sebuah metode untuk menangani masalah *stray node* telah dikemukakan sebelumnya. *Energy-Efficient-Multiple-Cluster-Head-Selection Routing-Protocol* (EEMCHRP) menangani masalah *stray node* (disebut *Lone Nodes* pada penelitian tersebut) dengan cara mengirimkan data menuju *Cluster Head Leader*. Hal tersebut dilakukan untuk mengurangi jumlah komunikasi langsung menuju *sink* yang boros energi (Noor-ul-Sabah, 2018). Namun dalam EEMCHRP model komunikasi *one-hop* tetap digunakan dan luas area yang disimulasikan relatif kecil yaitu 100x100m dengan kekuatan transimisi radio tiap *node* yang tidak terbatas.

2.3. *Location-based* GPSR

Location-based Greedy Perimeter Stateless Routing (GPSR) adalah sebuah algoritma *routing* yang berbasis lokasi (*location-based routing protocol*). Untuk mengimplementasikan *location-based* GPSR seluruh *node* pada jaringan harus mempunyai suatu metode untuk menentukan posisi geografisnya, sebagai contoh dengan menggunakan *Global Positioning system* (GPS). Lokasi geografis dari *node* inilah yang digunakan sebagai acuan untuk mengirimkan paket ke tujuan (Karp, 2000).

Pada *location-based* GPSR pesan akan diteruskan dengan mempertimbangkan jarak geografis yang terdekat untuk menuju lokasi tujuan pengiriman tanpa perlu untuk membangun sebuah *routing table*, oleh sebab itu algoritma ini disebut *stateless.* Jarak antara *node* dalam jaringan dihitung dengan menggunakan fungsi Eucledian berdasarkan koordinat 2-dimensi dari tiap *node* (Anasane & Satao, 2016).

Syarat digunakannya protokol *location-based* GPSR adalah:

- Masing-masing *node* mengetahui posisi geografisnya
- Masing-masing *node* mengetahui posisi geografis tetangga satu langkahnya

- Tujuan routing didefinisikan sebagai sebuah node pada koordinat posisi dua dimensi, atau sebuah area geografis yang direpresentasikan dengan *polygon* tertutup
- Setiap *packet* dapat membawa sejumlah kecil informasi *routing*

Protokol *location-based* GPSR dapat menemukan kegagalan jika terdapat *hole*, atau area dimana posisi pesan terakhir berada pada *node* yang paling dekat dengan *sink*, namun bukan merupakan *sink*, dan tidak ada tetangga yang lebih dekat dengan *sink* dibandingkan *node* terakhir pembawa pesan.

2.4. Energi Pada WSN

Dikarenakan keterbatasan sumber daya pada *node* penyusun jaringan WSN, energi adalah salah satu faktor penting yang perlu dibahas agar *network lifetime* dapat ditingkatkan. Secara umum *hardware* penyusun WSN adalah sebuah perangkat komputer berukuran kecil dengan kapasitas komputasi yang terbatas dengan fungsi untuk melakukan *sensing* dan komunikasi secara *wireless*.

Walaupun dengan perkembangan *microprosessor* yang semakin efisien dalam penggunaan energi, namun perkembangan teknologi baterai masih jauh tertinggal. Selain itu penggantian sumber energi pada *node* WSN mungkin tidak dapat dilakukan karena masalah biaya ataupun lingkungan yang ekstrim sehingga tetap diperlukan sebuah metode yang dapat menghemat energi pada WSN.

Pada model WSN tanpa kapabilitas *energy harvesting*, umumnya penghematan energi dilakukan dengan mengimplementasikan protokol-protokol baik untuk individu *node* dalam mengatur konsumsi energinya sendiri maupun untuk proses komunikasi antar *node* (protokol *routing)* agar energi dari jaringan dapat bertahan lebih lama (Rault, Bouabdallah, & Challal, 2014).

2.4.1. *Network Lifetime*

Network lifetime atau masa hidup sebuah jaringan sensor memiliki lebih dari satu definisi. Masing-masing definisi memiliki batasan dan belum ada yang bisa diterapkan pada semua kriteria jaringan. Beberapa penelitian awal merumuskan *network lifetime* sebagai rentang waktu sejak dimulainya transmisi data yang pertama hingga *node* terakhir mati (Tian, 2003), walaupun pada kenyataannya,

sebuah jaringan sensor sudah tidak bisa mengirim data hasil pemindaian, meskipun belum semua node mati.

Definisi berikutnya yang lebih realistis dikemukakan seiring dengan berkembangnya metode *routing* berbasis *cluster*. Soro dan Heinzelman (2005) menyatakan bahwa *network lifetime* adalah waktu hingga cluster head yang pertama mati. Namun definisi ini juga menjadi tidak relevan ketika protokol clustering mulai mampu menangani perubahan topologi dengan mengganti *cluster head*.

Pengertian lain dari *network lifetime* didefinisikan sebagai waktu jaringan mulai mengirimkan data hingga α persen dari total *sensor node* mati (Rajagopalan & Varshney, 2006). Dalam beberapa kasus pada jaringan WSN yang sangat mengutamakan lama waktu kemampuan operasi pada salah satu *sensor node*, *network lifetime* didefinisikan sebagai waktu *sensor node* yang pertama mengalami kehabisan energi untuk beroperasi. Definisi ini digunakan dalam penelitian ini dengan detail pengamatan yang dilakukan untuk evaluasi hasil adalah waktu saat *node* pertama mati, 20% *node* mati, dan 50% *node* mati dalam jaringan yang disimulasikan.

Network lifetime menjadi karakteristik utama pada saat melakukan evaluasi kinerja sebuah WSN. Bahkan ukuran *quality of service* dapat menurun dengan pertimbangan *network lifetime*. *Network lifetime* yang baik berpengaruh secara langsung dalam ketersediaan data hasil pemindaian dan penghematan biaya yang diperlukan untuk pemasangan baterai pada *sensor node* (Dietrich & Dressler, 2009)

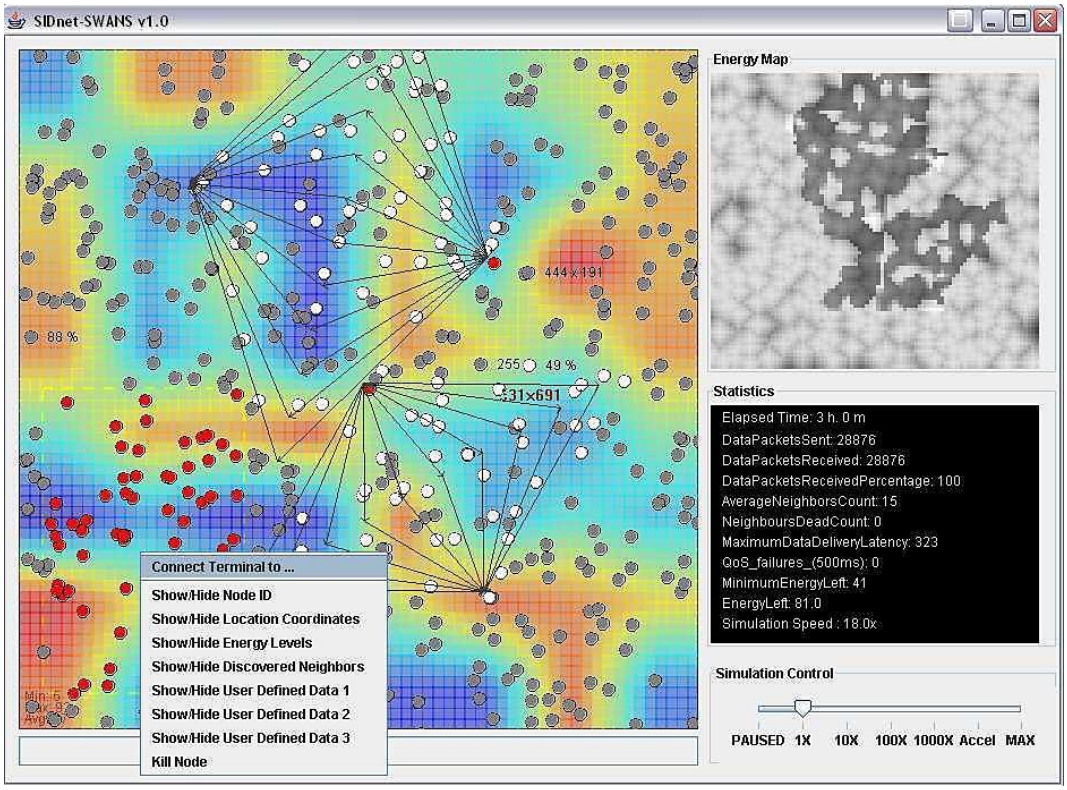
2.5. *Simulator and Integrated Development Platform for Sensor Networks Applications* (SIDnet-SWANS)

SIDnet-SWANS merupakan sebuah simulator jaringan WSN berbasis java yang paket distribusinya terdiri dari komponen JIST (*Java in Simulation Time*) dan SWANS (*Scalable Wireless Ad-hoc Network Simulator*). Simulator ini dipatenkan oleh Northwestern University pada tahun 2008 dengan penciptanya adalah Oliviu C. Ghica dkk. (Ghica, 2010). SIDnet dibuat untuk memberikan

simulasi dan *proof-of-concept platform* dimana pengguna aplikasi atau pengembang protokol *routing* dapat memantau fenomena pada jaringan sensor nirkabel yang berinteraksi.

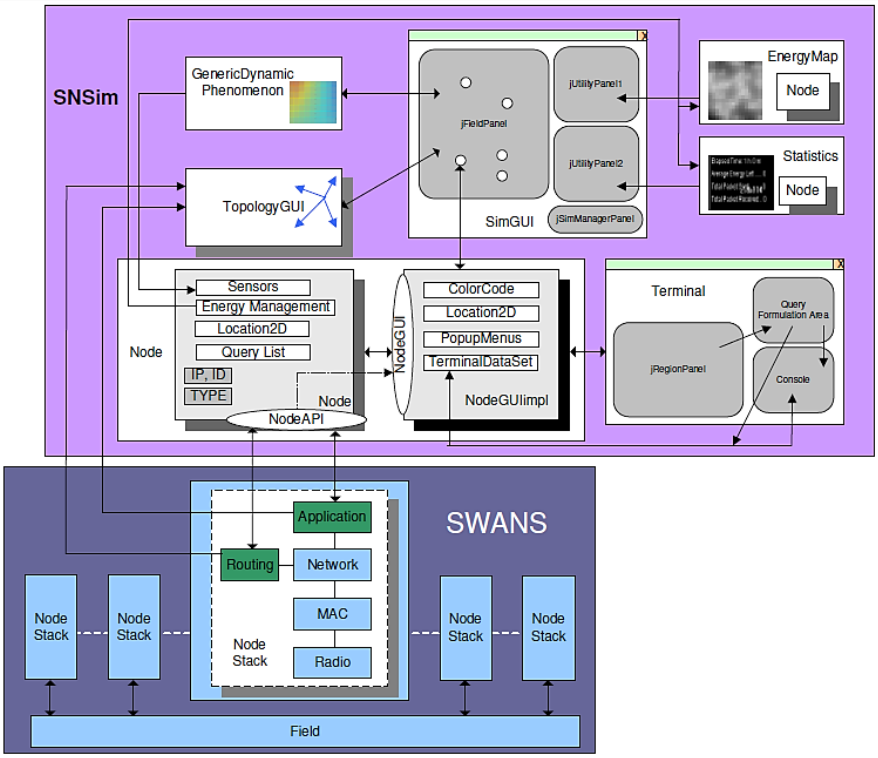
SIDnet adalah simulator yang berbasis Java yang dirancang untuk menunjukan interaksi pada jaringan. Pengguna dapat membuat dan mengamati algoritma baru yang dirancang dan mencoba mengaplikasikan skenario fenomena seperti apa yang dapat terjadi di kondisi nyata. Selain itu, pada saat *run-time* (melalui built-in terminal) waktu jalan simulasi bisa dipercepat atau dihentikan sementara sesuai kebutuhan pengguna. SIDnet menggabungkan antarmuka yang lengkap yang dibangun di atas simulator JiST-SWANS yang dapat menjamin validitas kinerja dan pengamatan dari simulasi tertentu. SIDnet adalah alat yang *extensible* yang dibuat dengan baik dan mudah untuk dipelajari.

Titik fokus pada simulator SIDnet-SWANS adalah *node* dimana SIDnet *node* merepsentasikan sebagai *interface* antara *network stack* dan seluruh komponen dari simulator termasuk GUI, *sensorial field*, lokasi *node,* manajemen energi dan lainnya (Ghica, 2010). Setiap aplikasi dan implementasi *network/routing* harus direfrensikan di setiap *node*. *Node* juga sebagai tempat penampungan informasi yang digunakan oleh seluruh elemen simulator seperti *neighboring node list*, tingkat energi dan lainnya.



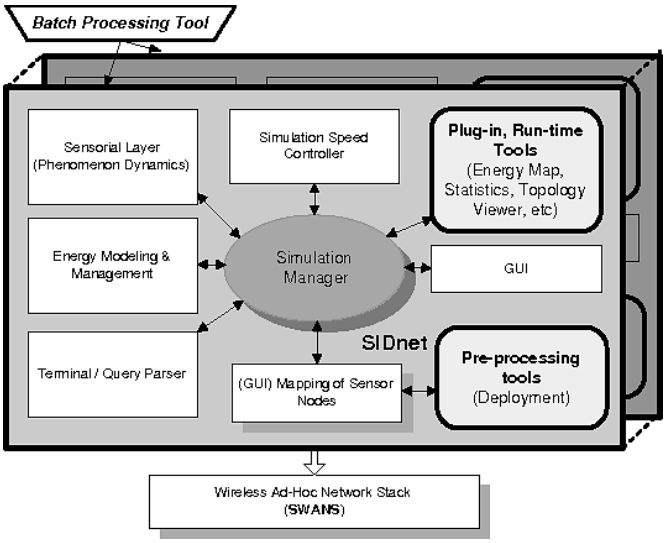
Gambar 2.4. Graphical User Interface (GUI) SIDnet-SWANS. (Ghica, 2010)

SIDnet-SWANS mensimulasikan 4 *network stack* yaitu *physical, Mac, Network,* dan *Application*; menghilangkan *Transport Layer* dari JiST SWANS karena dirasa tidak relevan dengan lingkungan WSN. SIDnet-SWANS dapat menampilkan proses simulasi dalam bentuk Graphical User Interface (GUI) seperti pada Gambar 2.4. Komponen penyusun SIDnet-SWANS adalah seperti pada Gambar 2.6 dan interaksi antara komponen *network stack* dan GUI adalah seperti pada Gambar 2.5.



Gambar 2.5. Arsitektur operasional SIDnet-SWANS. (Ghica, 2010)

Hardware yang akan disimulasikan dengan SIDnet-SWANS pada penelitian ini adalah Mica Mote MPR500CA dengan kapasitas baterai dan konsumsi energi adalah seperti pada Tabel 2.1.



Gambar 2.6. Komponen Penyusun SIDnet-SWANS. (Ghica, 2010)

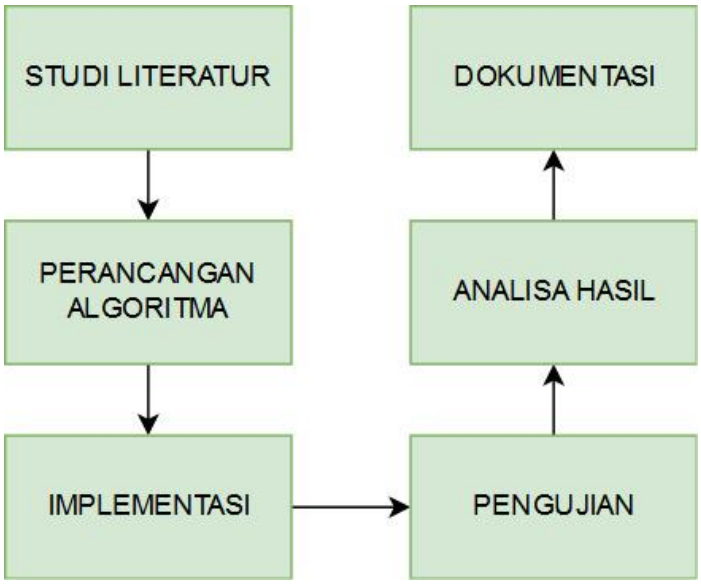
Tabel 2.1. Parameter Energi *Hardware* yang Disimulasikan.

Kapasitas baterai	75mAh
Processor Drain Active Mode	8mA
Processor Drain Sleep Mode	0.015mA
Radio Transmit Mode	27mA
Radio Receive Mode	10mA
Radio Listen Mode	3mA
Radio Sleep Mode	0.5mA
Sensor Active Mode	10mA
Sensor Passive Mode	0.01mA

BAB 3
METODE PENELITIAN

3.1. Tahapan Penelitian

Agar lebih terarah dan tercapainya tujuan yang diharapkan, penelitian ini akan dilakukan dengan beberapa tahapan yang ditunjukkan pada Gambar 3.1. Tahap awal yang akan dilakukan adalah studi literatur dengan tujuan untuk mempelajari masalah yang ada dan menemukan referensi yang dapat digunakan untuk landasan penelitian ini. Langkah kedua adalah perancangan algoritma untuk menyelesaikan masalah yang ditemukan pada proses studi literatur. Pada tahap ini akan dirancang algoritma untuk implementasi penanganan *multiple* CH dan *stray node* berbasis energi dengan tujuan meningkatkan performa algoritma LEACH terutama pada sisi *network lifetime*. Algoritma yang dibuat akan diimplementasikan pada simulator WSN SIDnet-SWANS untuk kemudian diuji dan diamati performanya. Hasil dari simulasi akan dianalisa dengan metrik *network lifetime*, PDR, *latency*, dan *residual energy* dari seluruh *node* yang disimulasikan. Tahap terakhir adalah proses dokumentasi dari proses dan hasil penelitian dalam bentuk tulisan berupa buku tesis.



Gambar 3.1. Diagram Alir Penelitian.

3.2. Studi Literatur

Studi literatur merupakan tahapan awal untuk melakukan kajian yang berkaitan dengan topik penelitian dengan mempelajari masalah dan menemukan referensi yang relevan. Referensi yang digunakan dalam penelitian ini berasal dari jurnal, konferensi dan buku yang berkaitan dengan algoritma LEACH pada lingkungan *Wireless Sensor Network*. Berdasarkan studi literatur yang telah dilakukan, diperoleh informasi sebagai berikut :

1. Karakteristik dan Keterbatasan WSN.
2. Perkembangan metode *routing* berbasis *cluster* pada WSN.
3. Pada metode *cluster* data sensing dikirim ke *cluster head* untuk kemudian dikirim menuju *sink.*
4. Algoritma routing LEACH membentuk *cluster* berdasarkan perbandingan *random number* dan perhitungan probabilitas yang dilakukan tiap *node.*
5. Timbul masalah *multiple CH* dan *stray node* pada algoritma *routing* LEACH jika digunakan parameter *hardware* sesuai dunia nyata.
6. Faktor energi perlu dipertimbangkan pada algoritma *routing* WSN untuk memaksimalkan performa jaringan.

3.3. Perancangan Algoritma

Perancangan algoritma yang dilakukan bertujuan untuk mengimplementasikan metode penanganan masalah *multiple* CH dan *stray node* pada algoritma routing LEACH. Awalnya setiap *node* akan di sebarkan secara acak pada luas area tertentu yang telah ditentukan dalam parameter simulator, kemudian *Heartbeat Protocol* akan dijalankan dengan tujuan mengetahui *one-hop neighbor nodes.* Setelah itu LEACH akan memulai membentuk *cluster* pada *setup phase*. Jika suatu *node* menjadi CH pada suatu *round* maka dia akan mengirim CH *advertisement message*. Bagi *node* yang tidak menjadi CH maka akan menunggu mendapatkan CH *advertisement message*, dan jika menerima pesan tersebut, maka node pengirim akan ditambahkan sebagai CH dan digunakan sebagi alamat *next-hop* untuk mengirim pesan. Jika setelah sebuah *node* mendapatkan CH dan menerima CH *advertisement message* lagi (*multiple* CH), maka akan

dibandingkan tingkat residual energy CH saat ini dengan tingkat energi CH baru dan akan dipilih CH dengan tingkat energi residual tertinggi.

Saat sebuah *node* tidak menjadi CH pada suatu *round* dan tidak juga menerima CH *advertisement message* (*stray node*), *node* tersebut akan mengirim *broadcast* pesan *stray advertisement.* Jika pesan tersebut diterima oleh *stray node* yang lain maka akan dibandingkan energi residual *node* tersebut dengan energi residual *node* pengirim pesan *stray advertisement*.

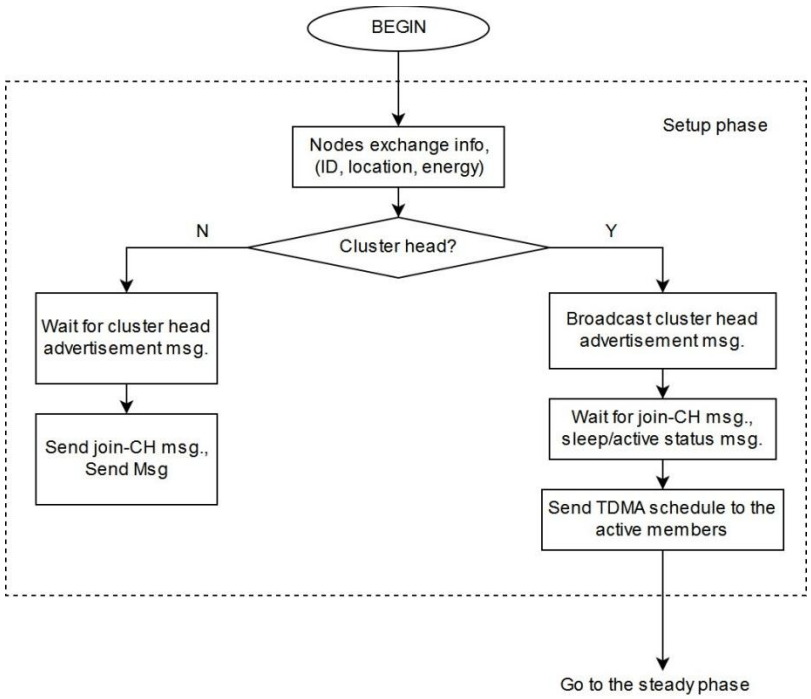
Setelah suatu stray node menerima beberapa pesan *stray advertisement* dan tingkat residual energinya adalah yang tertinggi, maka dia akan menjalankan fase *setup* LEACH ulang, dan jika dia terpilih menjadi CH maka akan mengirim CH *advertisement message* agar membentuk cluster dengan harapan dapat melayani para *stray node* disekitarnya dalam proses pengiriman data menuju *sink.*
Pada penelitian ini digunakan beberapa istilah, yaitu:

1. *Cluster head* : merupakan *node* yang bertugas menampung sementara dan meneruskan data dari *cluster member* ke *sink node*.
2. *Member Node* : merupakan *node* yang yang tidak terpilih menjadi *cluster head* pada suatu *cluster*.
3. *Stray Node*: merupakan *node* yang tidak menjadi CH dan tidak menerima CH *advertisement message* dalam suatu *round*.
4. *Cluster* : merupakan suatu area tertentu yang terdiri dari kumpulan beberapa member *node* yang memiliki *cluster head* yang sama.
5. *Data Message*: data *sensor* yang dihasilkan oleh *source node* yang merupakan hasil pengamatan terhadap lingkungan.
6. *Query Message* : permintaan yang dibuat oleh *sink* yang ditujukan ke semua *node* untuk melakukan pemindaian dengan interval waktu tertentu dan hasil pemindaiain tersebut akan dikirim ke *sink node.*
7. CH *Advertisement message* : pesan untuk memberi kabar pada *node* lainnya bahwa suatu node menjadi CH dan siap untuk menampung *data message* dari *member node* untuk di agregasi dan kemudian dikirim menuju *sink*.
8. *Stray Advertisement message* : pesan yang dikirim jika sebuah *node* menjadi *stray node* yang berfungsi sebagai dasar untuk membuat *cluster* dari *stray node*.

3.3.1. Implementasi LEACH

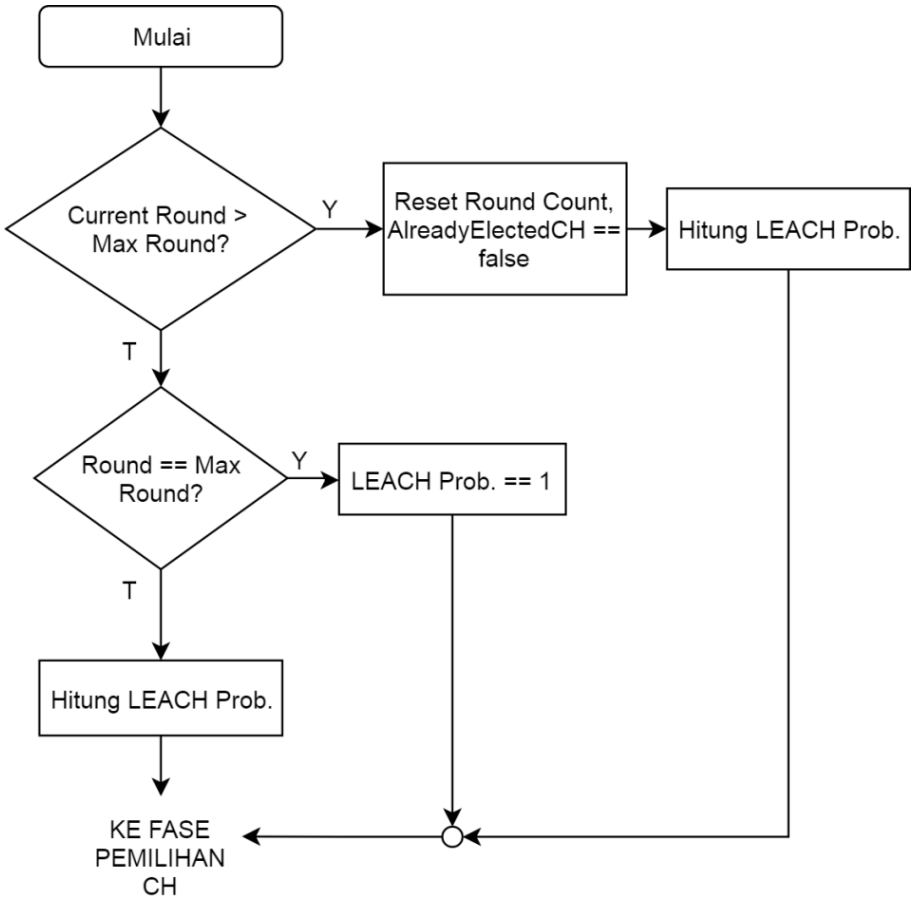
LEACH membentuk *cluster* pada fase *setup*. Tiap *round* baru seluruh *node* akan mengambil secara acak sebuah angka antara 1 dan 0, kemudian menghitung *threshold formula T(n).* Jika angka acak yang diambilnya kurang dari hasil *threshold* yang dihitung, maka *node* tersebut menjadi CH dan melakukan broadcast pesan bahwa dirinya dalah CH. *Node* yang tidak menjadi CH yang mendapat pesan tersebut akan bergabung kepada CH tersebut sehingga membentuk *cluster* (Raed & Abdalraheem, 2013). Formula yang digunakan ditunjukkan pada (3.1). Secara detail alur pembentukan *cluster* pada algoritma LEACH asli adalah seperti pada Gambar 3.2.

$$T(n) = \begin{cases} \frac{P}{1-P*(rmod\frac{1}{P})} if\ n \in G \\ 0, \qquad otherwise \end{cases} \tag{3.1}$$

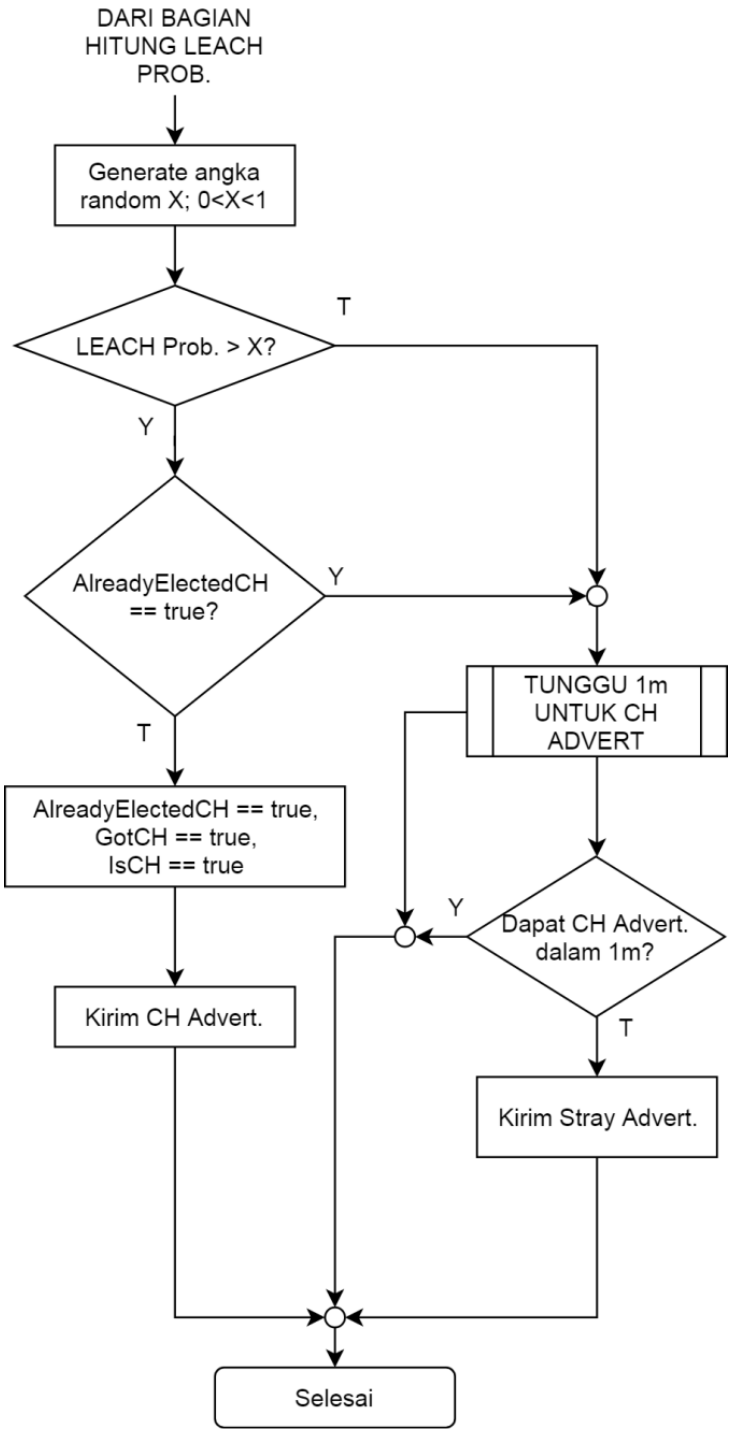


Gambar 3.2. Alur *Setup Phase* LEACH (Shukla, 2013).

Implementasi *round* pada penelitian ini dilakukan dengan cara memodifikasi pesan *heartbeat* dimana setiap 15 menit sekali seluruh *node* pada jaringan yang disimulasikan akan menjalankan ulang *setup phase* LEACH diikuti dengan *steady state*. Perhitungan waktu pada penelitian ini memanfaatkan parameter waktu simulasi global yang ada pada simulator SIDnet-SWANS. Perhitungan *round* yang digunakan dimulai dari *round* ke-0, dan nilai maksimum *round* tergantung dari nilai *P* yang digunakan yaitu 1/*P* (lihat (3.1)). Karena indeks *round* yang digunakan dimulai dari 0 maka *round* terakhir adalah *round* ke-(1/*P*-1). Secara umum jalannya *round* yang diimplementasaikan pada penelitian ini adalah seperti pada Gambar 3.3 dan fase *setup* seperti pada Gambar 3.4.



Gambar 3.3. Implementasi *Round* LEACH.



Gambar 3.4. Implementasi Fase *Setup* LEACH dengan Penambahan Penanganan *Stray Node*.

Pada Gambar 3.4, dapat dilihat terdapat sebuah mekanisme pengiriman *stray advertisement* jika suatu *node* tidak menjadi CH dan tidak mendapatkan pesan CH *advertisement message* pada suatu *round*.

3.3.2. Penanganan *Multiple CH* dan *Stray Node*

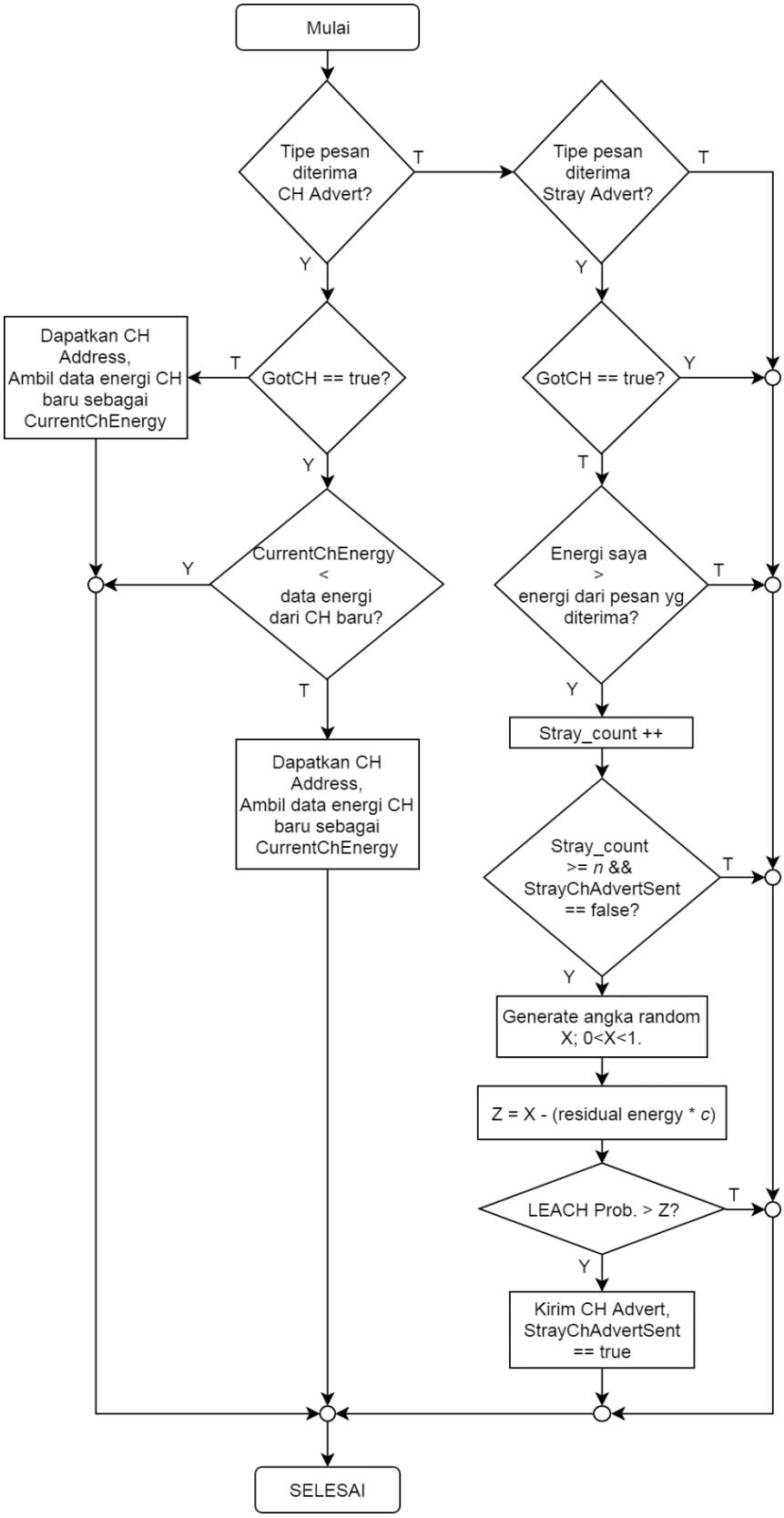
Implementasi penanganan *multiple* CH dan *stray node* akan dilakukan pada bagian kode yang menangani penerimaan pesan. Dalam penelitian ini digunakan 2 jenis pesan *advertisement* yaitu CH *advertisement message* (CH *advert*) dan *Stray advertisement message* (*Stray advert*). CH *advert* adalah pesan yang dikirim oleh sebuah *node* jika pada saat fase *setup* LEACH dia berhak menjadi CH. Isi pesan tersebut adalah alamat dari *node* pengirim, *residual energy node* pengirim, dan sebuah *flag* penanda bahwa pesan tersebut adalah hasil dari proses CH *election.*

Pesan CH *advert* jika diterima oleh sebuah *node* yang tidak menjadi CH dan belum memiliki CH pada suatu *round* maka *node* tersebut akan menyimpan alamat *node* pengirim pesan sebagai alamat CH-nya dan akan menjadikan *node* CH tersebut menjadi alamat *next-hop*-nya untuk keperluan mengirimkan pesan data yang didapatkan dari hasil *sensing*. Jika stelah memiliki CH sebuah *node* menerima pesan CH *advert* lagi dari *node* berbeda, maka akan dibandingkan *residual energy* dari CH baru dan CH yang telah dimilikinya dan akan dipilih CH dengan tingkat *residual energy* tertinggi.

Stray advert adalah pesan yang dikirim oleh *stray node*, yaitu *node* yang tidak menjadi CH dan juga tidak memiliki CH pada suatu *round*. Isi dari pesan tersebut adalah alamat dari *node*, sisa energi, dan juga sebuah penanda bahwa tipe pesan tersebut adalah pesan *stray advert*. Jika pesan tersebut diterima oleh *stray node*, maka node penerima tersebut akan membandingkan sisa energi dirinya dengan informasi energi yang terkirim pada pesan tersebut.

Jika tingkat energi *node* penerima lebih tinggi maka sebuah *counter* internal pada *node* tersebut akan di inkremen dan jika *counter* telah mencapai jumlah (*n),* yang menandakan bahwa telah ada sejumlah (*n) stray node* dengan tingkat energi yang lebih rendah daripada dirinya, maka akan dijalankan ulang proses *setup* LEACH dengan tujuan agar *node* tersebut dapat menjadi CH untuk para *stray node* disekitarnya. Jumlah *(n)* tetangga *stray node* yang akan dipertimbangkan nantinya akan diuji dan dianalisa terlebih dahulu pada bagian evaluasi untuk mendapatkan performa jaringan yang maksimal.

Pada saat proses LEACH *setup* ulang, hasil dari rangka acak X yang digunakan dalam proses komparasi dengan nilai *threshold* yang digunakan pada suatu *round* akan dikurangi dengan nilai *residual energy* dari *stray node* calon CH tersebut dikalikan dengan nilai konstanta *c*. Hal ini bertujuan agar jika *residual energy* dari *stray node* calon CH semakin tinggi, maka kemungkinan dirinya untuk menjadi CH juga semakin tinggi, dan pada akhirnya agar lebih tinggi kemungkinan dirinya untuk menjadi CH bagi para *stray node* disekitarnya. Nilai konstanta yang akan digunakan akan diuji untuk mendapatkan performa jaringan yang paling maksimal. Secara detail alur penanganan masalah *multiple* CH dan *stray node* dapat dilihat pada Gambar 3.5.

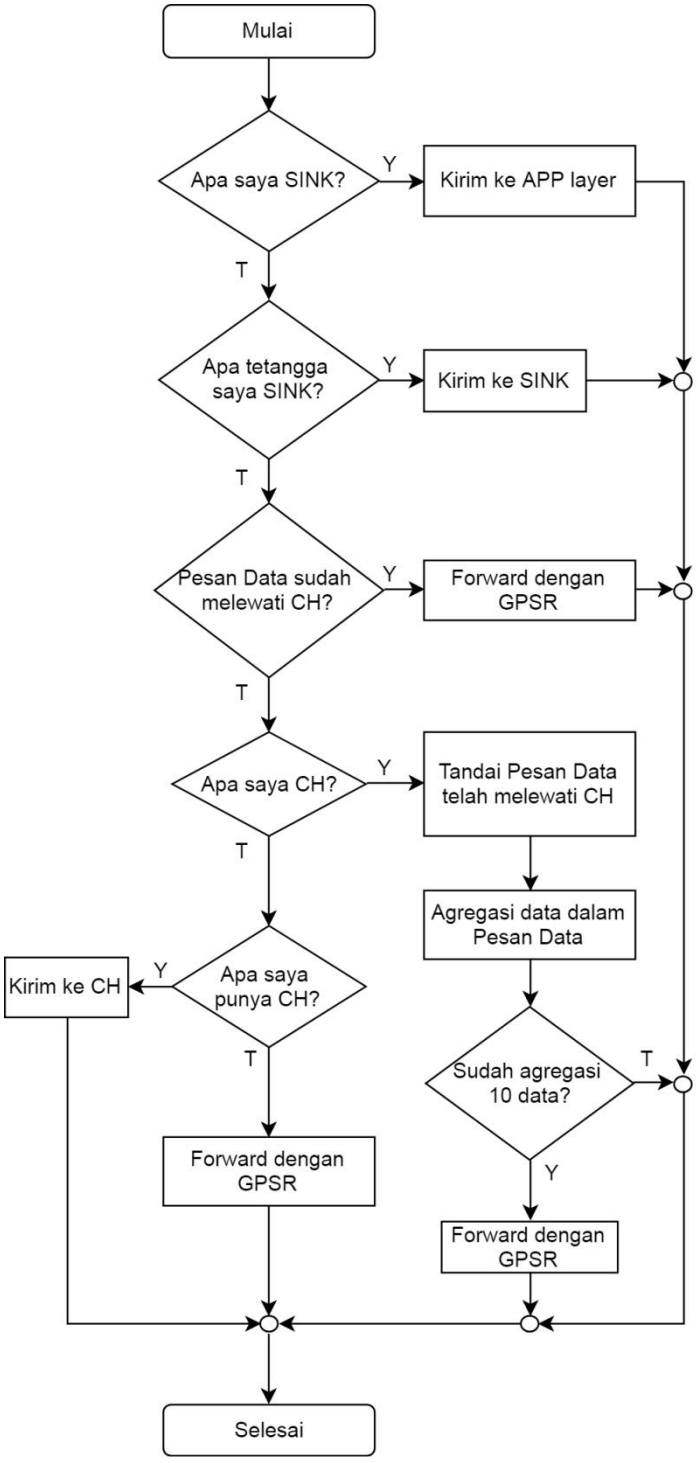


Gambar 3.5. Skema Penanganan Kasus *Multiple* CH dan *Stray Node*.

3.3.3. Fase *Steady State*

Pada fase ini dilakukan pengiriman data dari seluruh *sensor node* menuju *sink*. *Member node* akan mengirim data hasil *sensing* menuju CH-nya masing-masing dengan terlebih dahulu di agregasi untuk menghemat energi lebih lanjut. Sekumpulan *stray node* yang telah membentuk *cluster* sendiri maka juga akan mengirimkan data sensingnya pada CH barunya untuk di agregasi. Bagi *stray node* yang tetap menjadi *stray* selama suatu *round*, maka akan dia akan mengirim data dengan menggunakan GPSR, dan jika suatu saat data yang dikirimnya diterima oleh sebuah *member node*, maka data tersebut akan dikirimkan kepada CH dari *member node* tersebut untuk juga diagregasi.

Skala agregasi yang digunakan adalah 10:1, dimana untuk tiap 10 data hasil *sensing* yang diterima oleh CH, maka akan ada 1 data yang dikirim menuju *sink*. Pada akhirnya CH akan mengirim paket hasil agregasi tersebut menuju *sink* dengan model *multi-hop* menggunakan algoritma routing GPSR. Alur penerimaan dan pengiriman pesan data dapat dilihat pada Gambar 3.6.



Gambar 3.6. Alur Penerimaan dan Pengiriman Pesan Data.

3.4. Pengujian Algoritma

Implementasi dan pengujian algoritma akan dilakukan menggunakan simulator jaringan SIDNet-SWANS dengan beberapa ubahan pada parameter

algoritma guna mendapatkan performa algoritma yang maksimal dari sisi efisiensi penggunaan energi. Akan dibandingkan hasil evaluasi terhadap ubahan beberapa paremeter tersebut pada bagian pembahasan hasil penelitian.

3.4.1. Lingkungan pengujian

Dalam proses pengujian, algoritma yang dibuat akan diuji pada sebuah perangkat komputer. Spesifikasi dari komputer yang digunakan adalah sebagai berikut:

a. *Processor* Intel Core i7-4700HQ, 2.4 GHz
b. RAM 16 GB DDR3
c. *Harddisk* 1 TB 5400RPM
d. Sistem Operasi Windows 7 Ultimate x64
e. Java Development Kit 6

3.4.2. Parameter Uji

Parameter Uji yang akan digunakan pada penelitian ini bersifat tetap untuk proses implementasi dan evaluasi. Parameter uji ini diimplementasikan langsung pada *source code* simulator. Parameter uji yang digunakan dapat dilihat pada Tabel 3.1. Parameter *Radio* dan *Energy Consumption Paraameters* yang digunakan adalah sesuai *default* dari simulator SIDnet-SWANS yang mensimulasikan *hardware* nyata WSN yaitu Mica Mote MPR500CA.

Tabel 3.1. Parameter Uji.

No	Keterangan		Detail
1	Radio	Bandwidth (bps)	40000
		Transmit (dBm)	-12
2	Energy Consumption Parameters	ProcessCurrentDrawn_ActiveMode [mA]	8
		ProcessCurrentDrawn_SleepMode [mA]	0.015
		RadioCurrentDrawn_TransmitMode [mA]	27
		RadioCurrentDrawn_ReceiveMode [mA]	10
		RadioCurrentDrawn_ListenMode [mA]	3
		RadioCurrentDrawn_SleepMode [mA]	0.5
		*Sensor*CurrentDrawn_ActiveMode [mA]	10
		*Sensor*CurrentDrawn_PassiveMode [mA]	0.01
3	*Node Source*	Placement	Random
		Mobility	Static

		Sensor	GPS
		Battery Capacity (mAh)	40
		Battery Voltage (volt)	3
4	*Sink Node*	Placement	Grid (1,1)
		Battery Capacity (mAh)	80
		Battery Voltage (volt)	3
5	Sensing Query	Duration (hour)	70
		Sampling Interval (second)	3
6	Data Sensing	Random Value (Temperature)	18-39
7	Agregasi Data	Pada CH	10

3.5. Analisa Pengujian

Analisa pengujian sistem dilakukan dengan mengamati *packet delivery ratio, network lifetime*, dan *latency.* Peningkatan kinerja jaringan ditandai dengan meningkatnya *packet delivery ratio* dan *network lifetim*e, disertai dengan menurunnya *latency*. Untuk mengetahui peningkatan *packet delivery ratio* dan *network lifetime,* ditandai dengan menurunnya *traffic* jaringan *(packet drop)* dan semakin lama bertahannya *node* sensor di jaringan. Secara detail parameter evaluasi yang akan digunakan adalah sebagai berikut:

a. *Packet delivery ratio (PDR)*

PDR adalah perbandingan antara jumlah paket yang dikirim *source node* dan diterima oleh *destination node*. Karena pada penelitian ini digunakan agregasi maka PDR yang dihitung adalah yang berasal dari CH menuju *sink*. Rumus untuk menghitung PDR adalah seperti (3.3).

$$\text{PDR} = \frac{\sum Paket\ yang\ di\ Terima}{\sum Paket\ yang\ di\ Kirim}\ (\%) \qquad (3.3)$$

b. *Network lifetime*

Parameter *Network lifetime* yang diamati adalah waktu saat *node* pertama mati, 20% *node* mati, dan 50% *node* mati.

c. *Network average residual energy*

Adalah rata-rata sisa energi dari seluruh *node* dalam jaringan dalam suatu waktu tertentu, hal ini dapat dijadikan acuan tentang efisiensi sebuah algoritma routing dalam menggunakan energi.

d. *Latency*

Latency adalah rentang waktu yang diperlukan mulai paket dikirim hingga paket diterima oleh *sink*.

3.6. Skenario Evaluasi Kinerja

Evaluasi kinerja sistem ini dilakukan dengan cara mengamati dan menganalisa data yang diperoleh dari menjalankan algoritma *routing* yang dibuat dengan simulator SIDnet-SWANS. Pertama akan dilakukan evaluasi performa jaringan dalam masalah *multiple* CH dengan 2 penyelesaian yang berbeda, yang pertama dengan mempertimbangkan faktor jarak CH menuju *sink*, dimana CH dengan jarak terdekat menuju *sink* akan dipilih. Solusi yang kedua adalah dengan mempertimbangkan faktor *residual energy* dari CH dimana CH dengan *residual energy* tertinggi yang akan dipilih.

Untuk masalah *stray node* pertama akan dianalisa jumlah *stray node* yang harus dipertimbangkan pada proses pembentukan *cluster* dari *stray node*, setelah itu akan dievaluasi 3 hasil skenario dalam simulasi. Skenario yang pertama adalah jika tidak digunakan metode penanganan *stray node* dan pesan dari *stray node* akan secara langsung dikirimkan dengan menggunkan GPSR (Algoritma M-LEACH). Skenario yang kedua adalah jika digunakan metode penanganan *stray node* tanpa mempertimbangkan faktor energi, jadi sebuah *stray node* memiliki (*n)* tetangga *stray node* yang lain, maka dirinya akan langsung menjalankan ulang fase *setup* LEACH.

Skenario ketiga adalah jika digunakan metode penanganan *stray node* dengan mempertimbangkan faktor energi. Pada metode ini, sebuah *stray node* hanya akan menjalankan fase *setup* LEACH ulang jika dirinya memiliki (*n)* tetangga *stray node* yang tingkat energinya lebih rendah. Untuk dapat dilakukan perbandingan hasil maka jumlah *node* dan luas area yang digunakan dalam proses evaluasi adalah sama.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1. Tahapan Implementasi Metode

Pada tahapan ini akan dilakukan implementasi metode dengan menggunakan simulator yaitu SIDnet-SWANS. Implementasi yang dilakukan dengan memperbaharui modul-modul yang ada pada simulator SIDnet-SWANS, modifikasi dilakukan pada modul app, modul *routing*, modul *driver* dan modul *heartbeat protocol*. Tahapan implementasi metode penelitian yang digunakan adalah sebagai berikut:

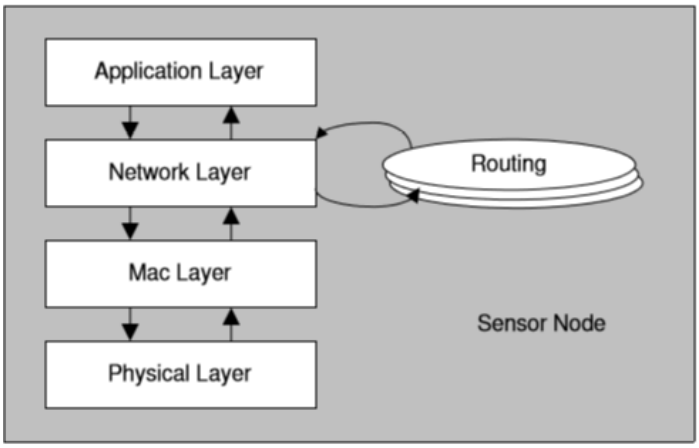
1. Tahapan pertama adalah melakukan perancangan terhadap protokol pengiriman untuk *wireless sensor network* dengan pengembangan mekanisme CH adoption dengan penanganan masalah *multiple* CH dan *stray node* dan akan diimplementasikan dengan menggunakan simulator berbasis java yaitu SIDnet-SWANS.
2. Tahapan kedua adalah pengujian yang akan dilakukan untuk menguji kinerja algoritma yang telah dihasilkan pada tahap pertama. Skenario yang dirancang meliputi variasi parameter pemilihan *cluster head* dan variasi jumlah *stray node* yang dipertimbangkan dalam pembentukan *cluster* dari *stray node*.
3. Tahapan ketiga adalah tahapan pengujian terhadap pengembangan mekanisme CH *adoption* dengan penanganan masalah *multiple* CH dan *stray node* berdasarkan parameter yang telah dirancang sebelumnya. Pada tahapan ini akan dihasilkan data hasil pengujian protokol yang telah di implementasikan dalam bentuk file teks.
4. Tahapan terakhir yaitu analisa hasil yang diperoleh pada tahapan sebelumnya, data hasil pengujian yang telah dilakukan akan disimpan dalam bentuk file teks yang kemudian akan dianalisa sesuai dengan parameter analisa yang telah dirancang sebelumnya.

Semua tahapan yang telah dijelaskan akan diimplementasikan menggunakan simulator SIDnet-SWANS dengan melakukan modifikasi pada modul *app*, modul

driver, modul *routing* dan *heartbeat*. Modul-modul tersebut berada pada *application layer* dan *network layer*, dimana kedua *layer* ini saling terhubung untuk bisa menyelesaikan suatu tugas. Hubungan antara kedua *layer* ini terlihat pada Gambar 4.1. Modifikasi dilakukan dengan cara menggandakan dan memodifikasi modul yang sudah ada ke dalam package sidnet.stack.users.CSGP.driver, sidnet.stack.users.LgSgp.*routing*, sidnet.stack.users.ChApp.app dan sidnet.stack.std.*routing*.*HeartbeatProtocol*

Application layer merepresentasikan setiap *node sensor* akan melakukan sensing sesuai dengan kriteria yang telah ditentukan oleh *sink* melalui pesan *heartbeat* dan mendapatkan data hasil *sensing* yang kemudian akan dikirim ke *sink*, untuk dapat mengirimkan data tersebut setiap *node* harus mengirimkan data menggunakan protokol tertentu. Proses pengumpulan informasi tetangga dan mendapatkan hasil sensing terjadi pada *application layer* dan kemudian data dan informasi tersebut akan dikirim ke *network layer* menggunakan *routing* tertentu. *Network Layer* merepresentasikan sebuah mekanisme *switchboard* antara paket yang datang dari *Application layer, Mac Layer*, dan paradigma *routing*. *Network Layer* meneruskan pesan berdasarkan alamat destinasi dari pesan tersebut.

Network layer akan memeriksa setiap pesan yang diterima apakah alamat dari pesan tersebut merepresentasikan tujuan akhir dari pesan tersebut. Jika tidak maka *network layer* akan meminta *routing protocol* untuk menentukan *"hop"* selanjutnya dan meneruskan pesan tersebut, akan tetapi jika alamat dari pesan tersebut merepresentasikan *node* tujuan akhir dari pesan tersebut maka *network layer* akan menangani pesan tersebut menggunakan *method receive* dan selanjutnya akan meneruskan pesan ke *application layer*.



Gambar 4.1. Hubungan antar *Layer* pada Simulator SIDnet-SWANS (Ghica, 2010)

4.1.1. Modifikasi pada Driver Simulator

Driver simulator adalah modul yang menggabungkan informasi pada masing-masing *layer* dan menjadi *entry point* pada aplikasi SIDNet-SWANS. Di dalam *driver* ditentukan kriteria *node*, topologi jaringan, algoritma yang dipakai pada masing-masing *layer*, statistik simulasi, dan lain-lain.

Modul *driver* pada simulator SIDnet-SWANS memiliki *class* utama, yaitu CSGPDriver.java yang menyatukan semua informasi terkait simulasi yang akan menjadi dasar parameter aplikasi SIDNet-SWANS. Modifikasi *method* createSim() dilakukan dengan penentuan parameter *stats collector*, yaitu parameter pemantauan statistik jaringan yang akan dikeluarkan sebagai *output* saat simulasi dijalankan.

Pada kelas CSGPDriver ditentukan jumlah *node*, luas area, kelas algoritma yang digunakan, kemampuan *node*, dan lain-lain. Dalam hal pembuatan *node*, modifikasi dilakukan pada method createNode(). Modifikasi method createNode() dilakukan pada *driver default* simulator untuk membuat *node* yang sesuai dengan algoritma *routing* yang diusulkan. Sebaran *node* ditentukan secara acak, sedangkan posisi *sink node* ditentukan pada koordinat (1,1) atau di pojok kiri atas dari area 2-dimensi yang disimulasikan. Pembuatan *node* beserta *properties*-nya dapat dilihat dalam *pseudocode* pada Gambar 4.2.

```
1   set node placement to random
2   set node mobility to static
3   set sink node placement to coordinate 1,1
4   set sink node mobility to static
5   add sensor
6   set sensor parameter to phenomenaLayer(temperature)
7   set node battery to 75 MJ
8   set node radio power to -12dBm
9   set sink node battery to 100 MJ
10  set sink radio power to -12dBm
```

Gambar 4.2. *Pseudocode* Pembentukan *Node* dan *Properties*-nya.

4.1.2. Modifikasi pada App Layer

Modifikasi *Application layer* dilakukan dengan cara memodifikasi kelas ChApp.java, MessageDataValue.java, dan MessageQuery.java. Pada file ChApp.java modifikasi dilakukan dengan menambah beberapa variabel global dan memodifikasi beberapa fungsi. Saat pertama berjalan ChApp akan menjalankan fungsi run() yang berisi program untuk mengirimkan pesan *heartbeat* yang fungsinya untuk mengetahui semua *node* tetangga yang berjarak *one-hop*. Fungsi yang di modifikasi pada ChApp terdapat pada bagian metode Sensing() yang berfungsi untuk melakukan pemantauan terhadap lingkungan dan fungsi Receive() yang berfungsi menerima pesan masuk yang berasal dari network layer. Pada fungsi Sensing() akan ditambahkan bagian *code* untuk menulis hasil pengamatan statistik jalannya simulasi pada *output terminal* untuk kemudian digunakan sebagai bahan analisa performa jaringan. Parameter *stats* yang dimaksud mengacu pada bagian deklarasi parameter statistik yang disebutkan pada kelas CSGPDriver.java.

Selain itu pada ChApp juga dilakukan proses pengambilan data *sensing* dan proses pengiriman data tersebut dengan *wrapper* tipe pesan MessageDataValue seperti yang dibuat pada kelas MessageDataValue.java. Kemudian pesan akan dikirimkan menuju *network layer*. Fungsi Receive() pada

ChApp digunakan untuk memproses pesan yang datang dari *network layer*, termasuk pesan *query* yang diinput oleh *user* pada GUI SIDnet-SWANS.

4.1.3. Implementasi LEACH dengan Metode Penanganan Multiple CH dan Stray Node

Implementasi algoritma yang diusulkan dilakukan dengan memodifikasi kelas *HeartbeatProtocol.java*. Pada kelas tersebut diberi penambahan implementasi *round* untuk keperluan LEACH pada *method* wakeAndBeat dimana setiap 15 menit sekali maka akan dijalankan ulang fase *setup* LEACH dengan variabel probabilitas P dan jumlah maksimal *round* yang telah ditentukan pada *hardcode*.

Pada bagian awal *method* wakeAndBeat() akan dilakukan pengecekan sisa energi dari *node*. Jika sisa energinya <5% maka dirinya akan mengirim pesan *unregister* untuk mengabari tetangganya bahwa dia sudah kehabisan energi. Setelah itu akan dilakukan pengecekan status jalannya *round* seperti yang telah didesain pada Bab 3. Modifikasi yang dilakukan untuk kedua fungsi tersebut adalah seperti pada Gambar 4.3.

Langkah selanjutnya akan dihitung *threshold* LEACH yang digunakan pada *round* tersebut. Setelah itu akan dijalankan proses komparasi angka *random* dengan hasil perhitungan *threshold* untuk menentukan apakah *node* menjadi CH apa tidak. Jika *node* menjadi CH maka akan dikirim pesan CH *advertisement* dengan cara *broadcast*. Isi dari pesan CH *advertisementi* adalah alamat dari *node*, sisa energi, dan jarak menuju *sink*. Informasi tersebut yang nantinya digunakan pada bagian penanganan masalah *multiple* CH pada bagian fungsi receive().

```
1   public synchronized void wakeAndBeat(long beatInterval,
2   boolean wakeAndBeatStarted)
3   {
..        if(myNode.getEnergyManagement().getBattery().getPer
4         centageEnergyLevel() < 5 && !unregistered){
5               //Node kehabisan energi akan mengirim pesan
6               Unregister
..              ...
7         }
..  ...
8         if (this.myNode.currentRound > this.maxround) {
9               //Menandakan akhir set round, maka parameter
10              akan di- reset, Hitung LEACH Threshold
..              ...
11        } else if (this.myNode.currentRound ==
12              this.maxround) {
13              //Menandakan round terakhir, set LEACH
14              Threshold menjadi 1
..              ...
15        } else {
16              //Menandakan bahwa round masih berjalan,
17              Hitung Leach Threshold
..              ...
18        }
..  ...
19  }
```

Gambar 4.3. Modifikasi yang Dilakukan untuk Implementasi *Round*.

```
1   public synchronized void wakeAndBeat(long
2   beatInterval, boolean wakeAndBeatStarted)
3   {
..  ...
4   this.currentRandom = Math.random();
5   if ((this.currentLeach > this.currentRandom) && (!
6   this.myNode.alreadyElectedCh))
7   {
8        //Set node menjadi CH, kirim pesan CH
9        Advertisement
..       ...
10       }
11  else {
12       //Set node tidak menjadi CH
..       ...
13       }
14  if(!this.myNode.hasGotCh && this.myNode.runSensing)
15  {
16       //Node tidak menjadi CH dan tidak dapat CH
17  setelah   1 min. Maka akan mengirim pesan Stray
18       Advertisement.
..            ...
19            }
..  ...
20  }
```

Gambar 4.4. Implementasi Proses Penentuan CH LEACH.

Jika suatu *node* tidak menjadi CH dan tidak mendapat CH (*stray node)* maka akan dikirimkan pesan *stray advert*. Implementasi modifikasi yang dilakukan adalah seperti pada Gambar 4.4.

```
1   public synchronized void wakeAndBeat(long
2   beatInterval, boolean wakeAndBeatStarted)
3   {
..  ...
4   public void receive(Message msg, NetAddress src,
5   MacAddress lastHop, byte macId, NetAddress dst, byte
6   priority, byte ttl)
7       {
..        ...
8         if (! (msg instanceof MessageHeartbeat)){
9               //Abaikan pesan karena tidak mengacu pada
10                    format yang dibuat
..              ...
11              }
12        if (((MessageHeartbeat)msg).isUnregistering()){
13              //Pesan penanda suatu node telah kehabisan
14                    energi, maka akan di-remove dari
15                    neighboursList
..              ...
16              }
..        ...
17      }
18  }
```

Gambar 4.5. Implementasi Penanganan Tipe Pesan Unregister dan Pesan yang Tidak Sesuai dengan Desain Algoritma.

Pada bagian *method* receive() pertama akan dilakukan penanganan penerimaan pesan *heartbeat* yang dikirim pada awal jalannya simulasi yang dihasilkan oleh ChApp. Jika pesan *heartbeat* diterima maka akan diambil informasi dari pesan tersebut untuk mengisi daftar tetangga node pada parameter NodeEntry. Isi dari NodeEntry adalah informasi tetangga dari suatu *node* berupa lokasi, *residual energy*, dan jumlah tetangganya. Selain itu pada bagian receive()

juga akan dipastikan tipe pesan yang masuk berdasarkan *flag* yang ada pada suatu pesan. Jika pesan yang masuk terdapat *flag unregister* maka *node* pengirim pesan tersebut akan dihapus dari daftar NodeEntry. Modifikasi kedua fungsi ini adalah seperti pada Gambar 4.5

```
1   public void receive(...)
2       {
..  ...
3   else if (((MessageHeartbeat)msg).isChElection() ){
4        if(!this.myNode.isClusterHead){
5             if ((this.myNode.hasGotCh) ... {
6                  //Komparasi energi CH baru dan CH
7                  sekarang, dipilih CH dengan energi
8                  tertinggi.
9                  }
10            else {
11                 //Tambahkan sebagai CH sekarang
12                 }
13            }
14       }
15       //Dia adalah CH, set next hop IP dia sendiri
..       ...
16  }
```

Gambar 4.6. Implementasi Penanganan Masalah Multiple CH.

Jika pada *method* receive() diterima pesan dengan *flag* isChElection, maka suatu *node* yang bukan CH dan belum mendapat CH akan mengambil informasi dari pesan tersebut berupa alamat dari CH dan residual energinya untuk disimpan pada variabel ChAdress dan ChBattery. Jika *node* penerima sudah memiliki CH maka akan dibandingkan antara *residual energy* yang didapat dari pesan CH *advert* dengan variabel ChBattery yang dimiliknya dan akan dipilih CH dengan tingkat *residual energy* tertinggi. Implementasinya ditunjukkan pada Gambar 4.6.

Jika pada *method* receive() diterima pesan dengan *flag* StrayAdvert dan dierima oleh *stray node* yang ditandai dengan parameter hasGotCh yang bernilai *false*, maka akan dibandingkan data *residual energy* dari pesan dengan sisa energi dirinya. Jika tingkat *residual energy* dirinya lebih tinggi dari *n* pesan *stray advert* yang diterimanya maka suatu *node* akan menjalankan ulang fase *setup* LEACH dengan nilai random X yang dikurangi dengan nilai (konstanta *c* **residual energy*) (lihat Gambar 3.5). Hal ini bertujuan agar *stray node* calon CH dengan nilai energi yang tinggi lebih tinggi pula kemungkinan dirinya agar menjadi CH untuk para *stray node* disekitarnya. Secara detail implementasi modifikasi untuk bagian penanganan *stray node* adalah seperti pada Gambar 4.7 dan dilanjutkan pada Gambar 4.8.

```
1    public void receive(...)
2        {
..   ...
3    else if (((MessageHeartbeat)msg).isStrayAdvert() ){
4              if(!this.myNode.hasGotCh &&
5         this.myNode.getEnergyManagement().getBattery().
6    get
7         PercentageEnergyLevel()>((MessageHeartbeat)
8         msg).getRemainingBattery()){
9              //Menandakan bahwa energinya lebih tinggi
10                  daripada stray node yang mengirim
..                  pesan.
12             this.currentStrayNeighbourCount += 1 ;
13             ...
```

Gambar 4.7. Implementasi Penanganan *Stray Node*.

```
14              if(this.currentStrayNeighbourCount >=
15                  15 && !this.strayChAdvertSent) {
16                  //Fase ulang setup, dengan pengaruh
17                  residual energy.
18                  this.currentRandom = Math.random();
19                   this.currentEnergyProbValue =
20      this.myNode.getEnergyManagement().getBattery().
21      getPercentageEnergyLevel() * c;
22          this.currentProbValue = this.currentRandom –
23          this.currentEnergyProbValue;
24                  }
25  if ((this.currentLeach >
26      this.currentProbValue) &&
27            (! this.myNode.alreadyElectedCh)){
28                  //Stray Node menjadi CH,Kirim CH
29                  Advertisement.
..                  ...
30                  }
31            }
32      }
..  ...
33  }
```

Gambar 4.8. Lanjutan Implementasi Penanganan Stray Node dari Gambar 4.7.

4.1.4. Modifikasi *Network Layer*

Implementasi algoritma dilakukan dengan memodifikasi kelas LgSgp.*java*. Pada kelas tersebut diberi penambahan implementasi penanganan tipe pesan data yang dihasilkan dari proses *sensing* pada *layer* aplikasi dalam *method* HandleMessageDataValue(). Implementasi dilakukan dengan acuan desain yang telah dibahas pada Bab 3.

```
1   private void handleMessageDataValue(NetMessage msg,
2   Location2D targetLocation) {
..  ...
3       if((this.myNode.getIP() == SinkIp)) {
4       //Jika dirinya sink maka dikirim ke app layer
..      ... }
5       else if(myNode.neighboursList.contains(SinkIp)){
6       //Jika tetangganya sink, maka langsung dikirim
7       ke sink
..      ... }
8   if(!msgAGW.hasPassedCH()&&(myNode.isClusterHead) {
9       //Agregasi 10 data, lalu diteruskan dengan GPSR,
10      kirim ke link layer
..      ...
11      }
12      else if ((!msgAGW.hasPassedCH() &&
13            (!myNode.isClusterHead))){
14      //Diteruskan dengan GPSR, kirim ke link layer
..      ...
15      }
```

Gambar 4.9. Implementasi Penanganan Tipe Pesan Data.

Pada *method* HandleMessageDataValue() semua kemungkinan status pesan data yang diterima akan dicek dan pesan akan diteruskan sesuai dengan statusnya masing-masing. Jika sebuah pesan akan diteruskan dengan GPSR, maka alamat *next-hop* yang dipakai didapatkan dari fungsi getThroughShortestPath(). Secara detail implementasi penanganan tipe pesan data dijelaskan pada Gambar 4.9 dan Gambar 4.10.

```
16      else { //Diteruskan dengan GPSR
17      ... }
..  ...
18      if (this.myNode.hasGotCh){ //Diteruskan ke CH
..      ...
19      }
20      else { //No CH, diteruskan dengan GPSR
..      ...
21      }
22      if ((msgAGW.hasPassedCH()==false) &&
23      (!this.myNode.hasGotCh)){
24      //Diteruskan dengan GPSR, kirim ke link layer
25      ...}
26      else  if ((msgAGW.hasPassedCH()==false) &&
27      (this.myNode.hasGotCh)){
28           if(nextHop== myNode.getIP()){
29           //Dia sendiri CH, pesan di agregasi, kirim
30                dengan GPSR jika sudah 10
..           ...
31           }
32           else {
33           //pesan akan dikirim ke CHnya
34           ... }
35      }
36      else if(msgAGW.hasPassedCH()==true){
37      //Pesan  pengiriman  hasil  agregasi  CH,  kirim
38      dengan GPSR
39      ... }
..  ...
40  }
```

Gambar 4.10. Implementasi Penanganan Tipe Pesan Data Lanjutan dari Gambar 4.9.

4.2. Tahapan Uji Coba

Pengujian dilakukan untuk mengetahui kinerja protokol usulan yaitu pengembangan mekanisme CH *adoption* dengan penanganan masalah *multiple* CH dan *stray node*. Ujicoba yang dilakukan untuk mengetahui parameter pemilihan *cluster head* yang optimal. Pengujian yang dilakukan bertujuan untuk mengetahui kinerja *penanganan masalah multiple CH dan stray node* jika dibandingkan dengan metode tanpa penanganan *stray node* sehingga dapat mengetahui perbandingan kinerja masing-masing protokol sesuai dengan skenario yang telah dipaparkan sebelumnya.

Langkah –langkah pengujian pada penelitian ini adalah dengan membuat skenario pengujian, menentukan parameter pengujian dan menganalisa hasil pengujian yang telah dilakukan.

4.2.1. Skenario Pengujian

Untuk mengetahui efektifitas metode yang diusulkan, seluruh metode diuji akan diamati performanya dengan menggunakan nilai *P* LEACH yang berbeda (lihat (3.1)). Rentang nilai *P* yang digunakan adalah [0.04-0.09] dengan inkremen 0.01. Hal ini dilakukan karena nilai *P* yang digunakan akan mempengaruhi jumlah *cluster* yang terbentuk pada LEACH dan akan mempengaruhi total panjang *round* dan jumlah *multiple* CH dan *stray node* pada jaringan.

Jika nilai *P* yang digunakan rendah maka total *round* akan semakin panjang, namun angka *threshold* LEACH akan dimulai dengan hasil yang rendah sehingga pada beberapa *round* awal akan sedikit kemungkinan terbentuk CH sehingga akan lebih banyak *stray node*. Sebaliknya jika nilai *P* yang digunakan semakin tinggi maka total *round* akan semakin sedikit dan *threshold* LEACH akan dimulai dengan hasil yang tinggi sehingga akan lebih banyak CH yang muncul sehingga lebih mungkin masalah *multiple* CH akan terjadi.

Untuk penanganan *stray node* akan diuji beberapa variasi jumlah *stray node* yang harus dipertimbangkan saat pembentukan *cluster* dari *stray node* seperti pada Tabel 4.1., dan juga diuji beberapa nilai konstanta *c* yang akan digunakan.

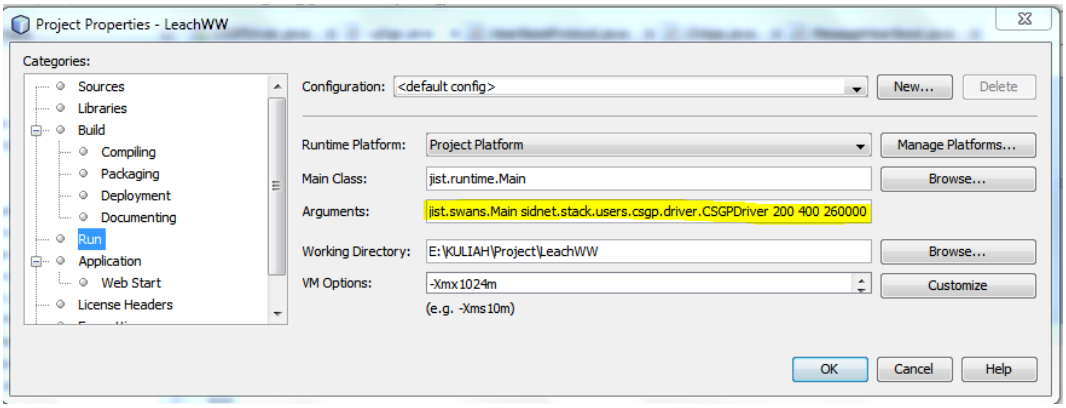
Tabel 4.1. Persentase dan Jumlah *Node* yang Dipertimbangkan Pada Proses Penanganan *Stray Node* untuk Dianalisa.

Persentase (dari 200 node yang disimulasikan)	**Jumlah Node**
2,5	5
5	10
7,5	15
10	20
12,5	25

4.2.2. Parameter Pengujian dan Eksekusi Simulator

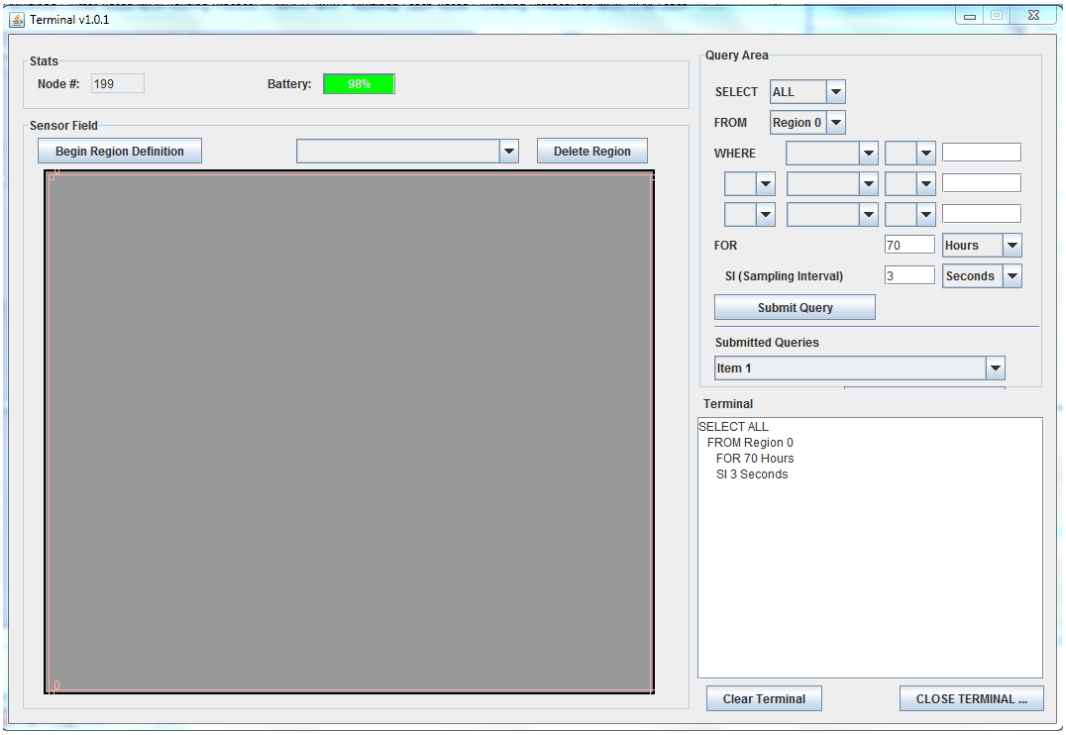
Parameter eksekusi simulator ditentukan dalam kelas idnet.stack.users.CSGP.driver. Kelas ini berfungsi sebagai modul yang menggabungkan informasi pada masing-masing *layer* dan menjadi *entry point* pada aplikasi SIDnet. Di dalam driver ditentukan kriteria *node*, topologi jaringan, algoritma yang dipakai pada masing-masing *layer*, statistik simulasi, dan lain-lain. Modul driver pada simulator SIDnet-SWANS memiliki *class* utama, yaitu CSGPDriver.java yang menyatukan semua informasi terkait simulasi dan menjadi aturan dasar menjalankan aplikasi SIDNet-SWANS.

Parameter jumlah *node*, waktu berhenti simulasi dan luas area akan dieksekusi oleh simulator dengan memasukkan parameter pengujian tersebut melalui *run arguments* yang dapat diakses pada fitur *project properties* di NetBeans sebagaimana divisualisasikan pada Gambar 4.11., parameter ini yang nantinya akan digunakan kelas *driver* sebagai acuan jalannya simulasi. Dari gambar tersebut dapat dilihat bahwa kelas *driver* yang dipakai adalah pada direktori jist.swans.Main sidnet.stack.users.csgp.driver.CSGPDriver, dengan jumlah *node* yang disimulasikan 200, luas area 400x400m, dan waktu berhenti simulasi 260000s atau 72.2 jam. Waktu berhenti simulasi tersebut dipilih karena dalam pengujian akan digunakan total waktu *sensing* selama 70 jam dan waktu menjalankan *heartbeat* hingga awal *sensing* dilakukan adalah 1 jam.



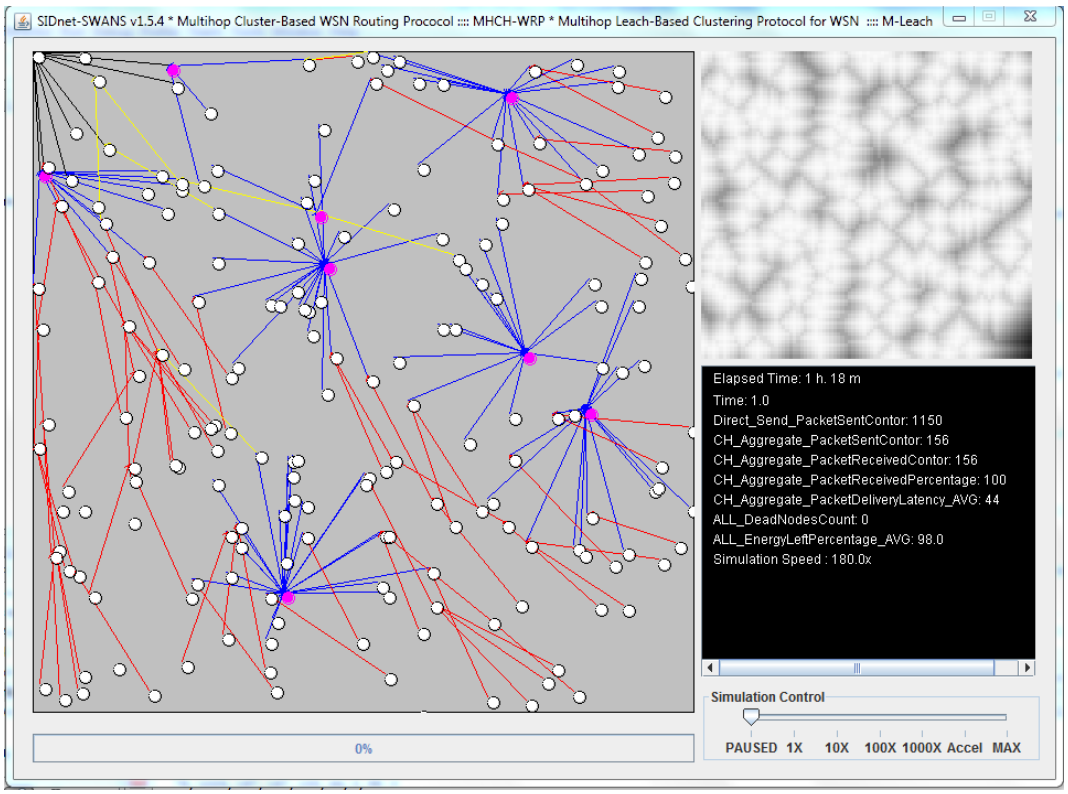
Gambar 4.11. *Run Arguments* yang Digunakan Dalam Simulasi.

Eksekusi program dilakukan dengan memilih menu *run* pada IDE NetBeans. Simulator akan menampilkan tampilan GUI yang dapat digunakan untuk melakukan interaksi dengan *node*. Selama satu jam pertama, simulator akan menjalankan protokol *heartbeat* dan waktu simulasi dipercepat hingga seribu kali. Pada menit ke-59, waktu simulasi dilambatkan menjadi *real-time* untuk memberi kesempatan berinteraksi dengan simulator.



Gambar 4.12. Jendela Input *Sensing Query* SIDnet-SWANS.

Node yang dipilih untuk menjadi *sink* diberi perintah *query* dengan cara membuka aplikasi terminal simulator melalui *node* tersebut seperti pada Gambar 4.12. Parameter pengujian meliputi batas waktu simulasi, interval pemindaian, area pemindaian, dan batas waktu pengamatan dapat dispesifikasikan melalui aplikasi terminal untuk kemudian dikirim sebagai *query message* sebagai pedoman parameter jalannya simulasi untuk semua *node* pada jaringan. Visualisasi jalannya simulasi SIDnet-SWANS ditampilkan melalui GUI seperti pada Gambar 4.13.



Gambar 4.13. Visualisasi Jalannya Simulasi pada SIDnet-SWANS.

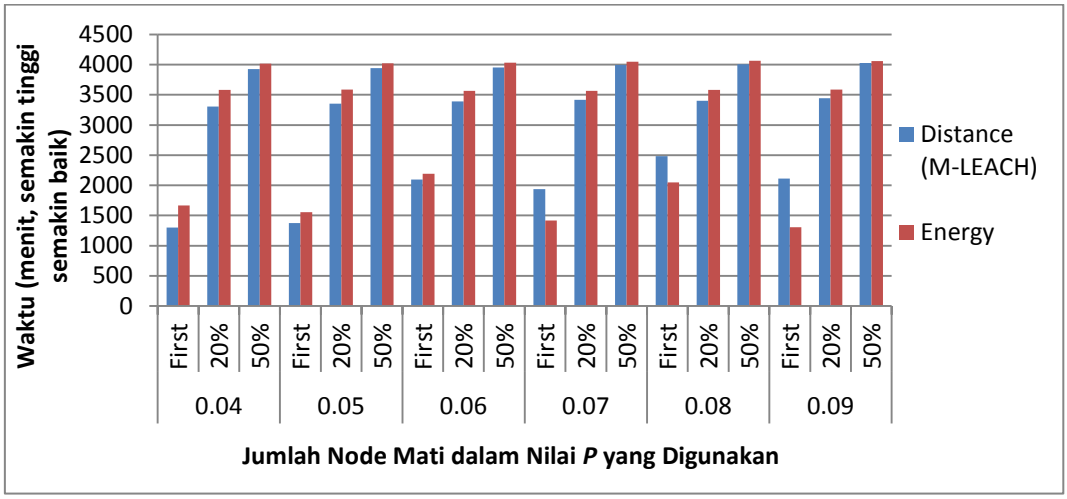
4.3. Hasil dan Analisis

Hasil pengujian pada sub bab ini diperoleh dengan melakukan analisis terhadap file teks yang dihasilkan saat algoritma *routing* yang dibuat dijalankan pada SIDnet-SWANS. Dikarenakan sifat alami LEACH yang cenderung *random* maka untuk mendapatkan hasil yang *valid* tiap skenario simulasi dijalankan masing-masing 3 kali *run* dan diambil nilai rata-ratanya untuk dianalisa.

4.3.1. Analisa Penanganan Masalah *Multiple CH*

Dalam menangani masalah multiple CH, 2 parameter pemilihan CH yaitu jarak menuju *sink (distance)* (M-LEACH) dan *residual energy (energy)* diimplementasikan dan dianalisa. Jika parameter *distance* digunakan, maka dalam kasus *multiple* CH akan dibandingkan jarak dari CH menuju *sink* dan dipilih CH dengan jarak paling dekat dengan *sink*. Hal ini bertujuan agar sebisa mungkin mengurangi jumlah *hop* yang ditempuh saat paket akan dikirimkan menuju *sink*.

Jika parameter *energy* digunakan maka akan dipilih CH dengan kapasitas *residual energy* yang paling tinggi. Hal ini bertujuan agar pekerjaan CH yang lebih berat dari sisi penggunaan energi akan dilakukan oleh CH dengan energi tertinggi. Dari hasil pengujian dan analisa yang dilakukan didapati bahwa dengan menggunakan parameter *energy* dalam kasus *multiple* CH, hasilnya secara keseluruhan lebih baik dari sisi *network lifetime*. Hal ini ditandai dengan waktu *node* yang mati pertama, 20%, dan 50% dalam jaringan relatif lebih lama jika dibandingkan dengan menggunakan parameter *distance*. Secara detail hal ini ditunjukkan pada Gambar 4.14.



Gambar 4.14. Perbandingan Waktu Node Mati antara Penggunaan Parameter *Energy* dan *Distance* pada Masalah Multiple CH.

Hasil simulasi yang dilakukan juga menunjukkan bahwa dengan mempertimbangkan parameter *energy* dalam kasus *multiple* CH menghasilkan

total *packet sent* dan *received*, serta PDR yang rata-rata lebih bagus dibandingkan dengan penggunaan parameter *distance* pada semua nilai *P* yang digunakan dalam simulasi. Hal ini berbanding lurus dengan *residual energy* jaringan yang lebih tinggi, dimana semakin lama *node* aktif makajumlah paket yang di-*generate* akan semakin banyak dan kecenderungan paket diterima oleh tujuan akan semakin tinggi. Secara detail hasil perbandingan jumlah paket yang dikirim dan diterima ada pada Tabel 4.2, dan perbandingan PDR ada pada Tabel 4.3.

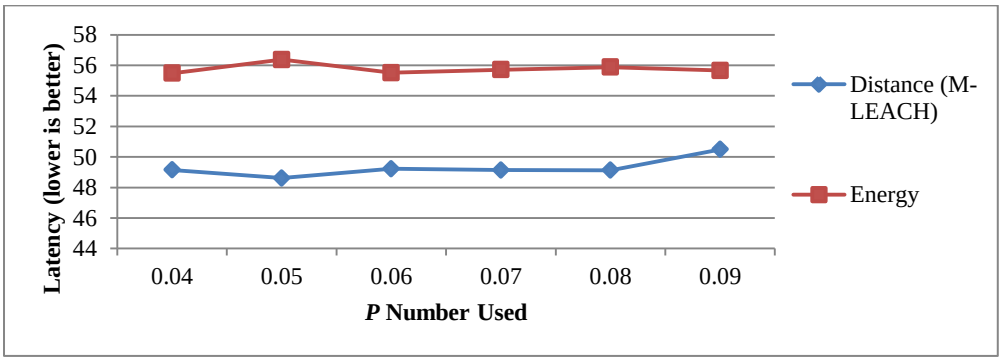
Tabel 4.2. Perbandingan Jumlah Total Paket Dikirim dan Diterima dari Dua Parameter Penanganan *Multiple* CH yang Disimulasikan.

	NILAI *P* YANG DIGUNAKAN											
	0.04		0.05		0.06		0.07		0.08		0.09	
	Sent	Recv.	Sent	Recv.	Sent	Recv.	Sent	Recv.	Sent	Recv.	Sent	Recv.
DIST. (M-LEACH)	207741	168315	211296	168078	215008	179471	217118	177020	218995	178886	221646	185560
ENER-GY	231690	192956	234806	198746	236030	192757	238510	202584	239955	204477	239646	203099

Tabel 4.3. Perbandingan PDR dari Dua Parameter Penanganan *Multiple* CH yang Disimulasikan.

	NILAI *P* YANG DIGUNAKAN					
	0.04	0.05	0.06	0.07	0.08	0.09
DISTANCE (M-LEACH)	**81%**	**80%**	**83%**	**82%**	**82%**	**84%**
ENERGY	**83%**	**85%**	**82%**	**85%**	**85%**	**85%**

Dari Gambar 4.15 dapat dilihat bahwa dengan menggunakan parameter *distance* sebagai faktor pertimbangan pemilihan CH pada masalah *multiple* CH menghasilkan rata-rata *latency* yang lebih rendah. Hal ini terjadi karena jika digunakan parameter *distance*, CH yang dipilih oleh *member node* akan selalu yang terdekat dengan *sink* sehingga jarak yang ditempuh sebuah paket saat menuju *sink* akan lebih pendek. Namun untuk selanjutnya dalam penelitian ini akan digunakan faktor energi dalam pertimbangan memilih CH pada masalah *multiple* CH karena dari sisi penggunaan energi hasilnya paling efisien.



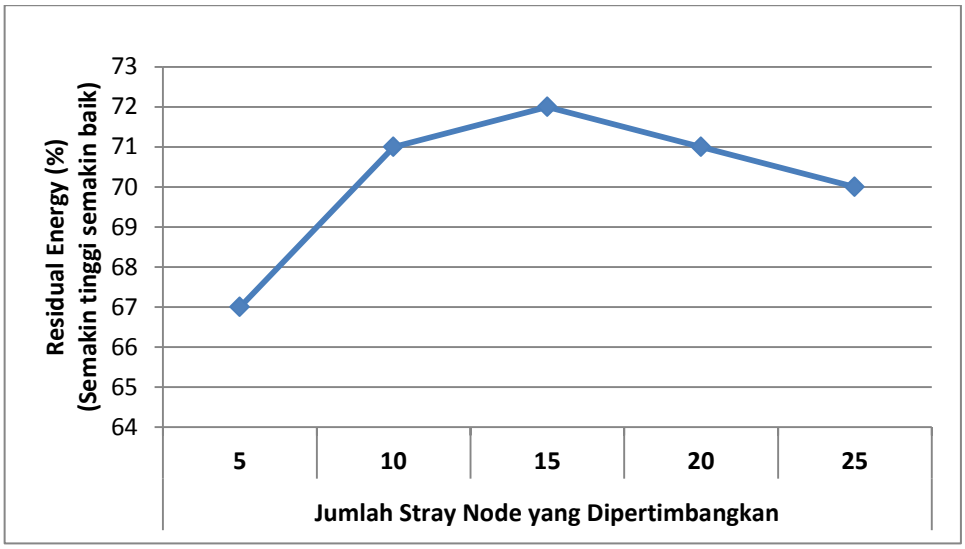
Gambar 4.15. Perbandingan *Latency* antara Penggunaan Faktor *Energy* dan *Distance* dalam Masalah *Multiple* CH.

4.3.2. Hasil Penanganan Masalah *Stray Node*

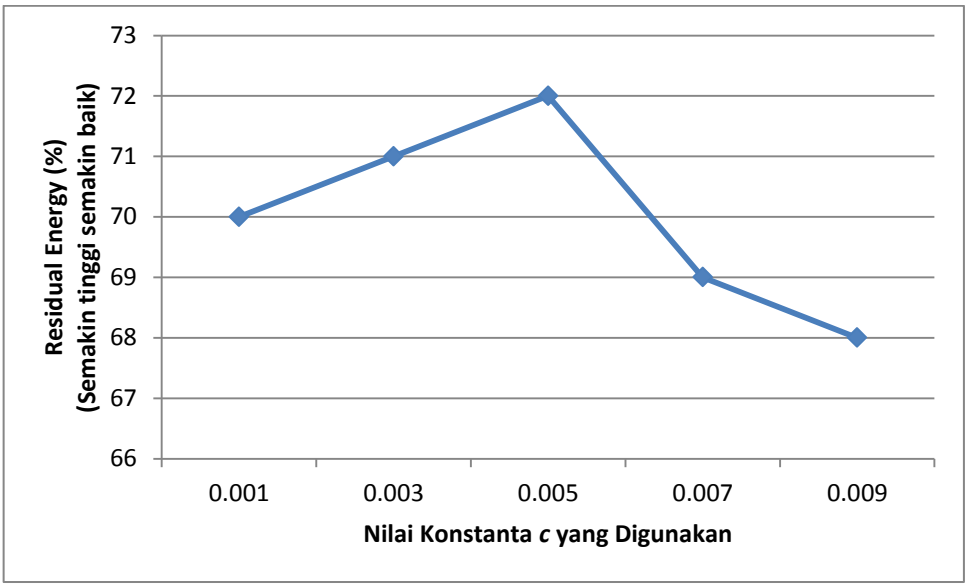
Dalam penanganan masalah *stray node* akan dianalisa hasil dari skenario evaluasi yang telah dirancang pada Bab 3. Pertama, hasil analisa jumlah *stray node* yang dipertimbangkan pada proses pembentukan *cluster* dari *stray node* dapat dilihat pada Gambar 4.16. Dari gambar tersebut dapat dilihat bahwa dengan menggunakan jumlah 15 *stray node* yang harus dipertimbangkan pada saat proses pembentukan *cluster* dari *stray node* menghasilkan *residual energy* yang paling tinggi. Oleh sebab itu pada bagian selanjutnya pada penelitian ini akan digunakan jumlah tersebut.

Selanjutnya diuji nilai konstanta *c* yang digunakan dalam proses LEACH ulang bagi *stray node* calon CH. Dari gambar Gambar 4.17 dapat dilihat bahwa dengan menggunakan nilai konstanta $c = 0.005$ menghasilkan *residual energy*

yang paling tinggi pada T-1000m. Oleh sebab itu nilai tersebut akan digunakan pada bagian selanjutnya.



Gambar 4.16. *Residual Energy* pada T-1000m untuk Jumlah Pertimbangan *Stray Node* yang Berbeda pada saat Pembentukan *Cluster* dari *Stray Node*.

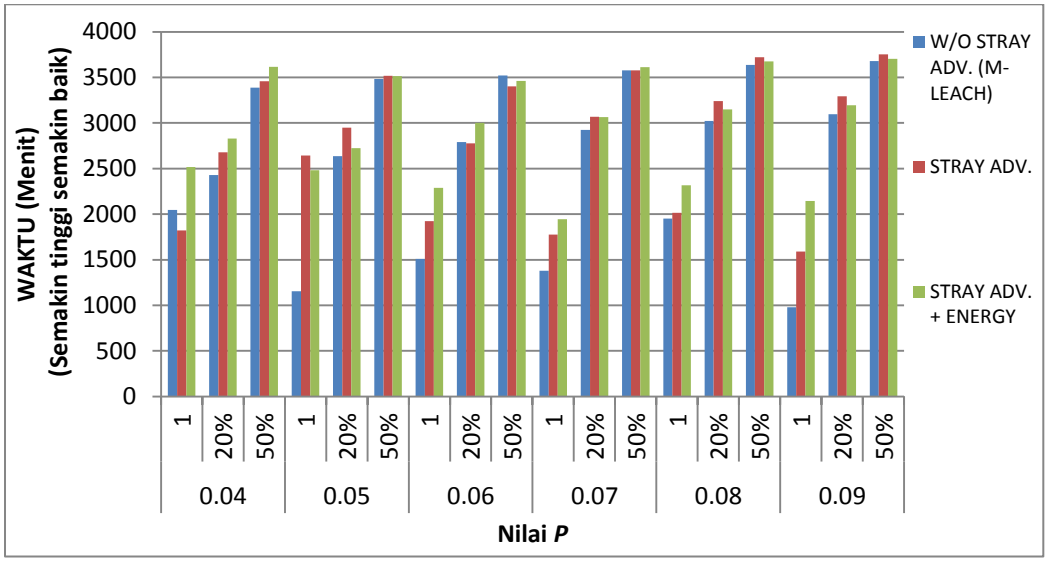


Gambar 4.17. *Residual Energy* pada T-1000m untuk Nilai Konstanta *c* yang Berbeda pada saat Fase LEACH *Setup* Ulang.

Untuk analisa selanjutnya adalah perbandingan 3 hasil simulasi yaitu jika tidak digunakan metode penanganan *stray node* (M-LEACH), jika digunakan metode penanganan *stray node* tanpa mempertimbangkan faktor energi, dan jika

digunakan metode penanganan *stray node* dengan mempertimbangkan faktor energi. Dari

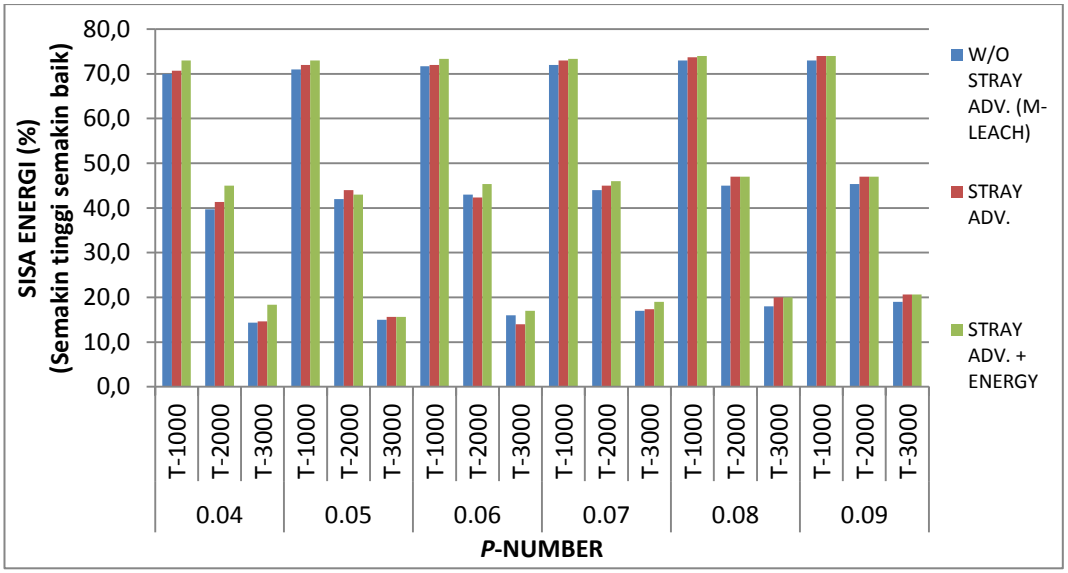
Gambar 4.18. dapat disimpulkan bahwa dengan menerapkan metode penanganan *stray node* dengan pertimbangan energi menghasilkan rata-rata *network lifetime* yang paling tinggi dalam semua nilai *P* yang disimulasikan. Hal ini ditunjukkan dari waktu *node* pertama mati yang paling tinggi yang berarti *node* dalam jaringan semakin hemat dalam penggunaan energinya. Selain itu waktu 20% dan 50% *node* mati pada jaringan cenderung lebih tinggi dibandingkan dengan 2 skenario yang lain terutama pada penggunaan nilai *P* yang rendah dimana akan lebih banyak *stray node* yang muncul.



Gambar 4.18. Perbandingan Waktu *Node* Mati terhadap Waktu dalam Nilai P Berbeda.

Pernyataan ini diperkuat dengan hasil analisa *residual energy* yang ditampilkan pada Gambar 4.19. Dari gambar tersebut menunjukkan bahwa dengan menerapkan metode penanganan *stray node* dengan pertimbangan energi menghasilkan nilai *residual energy* yang rata-rata paling tinggi untuk semua *sampling* waktu yang diamati yaitu T-1000, T-2000, dan T-3000 menit pada semua nilai *P* yang digunakan dalam simulasi. Hal ini terjadi karena dengan mempertimbangkan faktor *residual energy* dari *stray node* maka *node* yang

berhak menjadi CH untuk para *stray node* adalah *node* yang memiliki *residual energy* yang paling tinggi sehingga pekerjaan CH yang menyebabkan konsumsi energi ekstra akan jatuh pada *node* yang lebih sesuai.

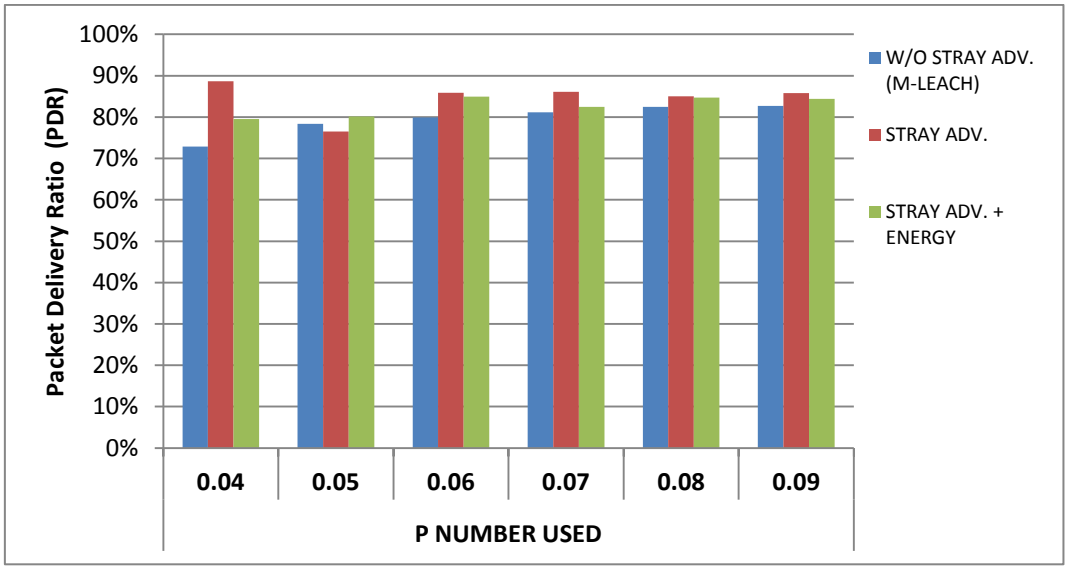


Gambar 4.19. *Residual Energy* Rata-rata Hasil Pengujian yang Dilakukan.

Berdasarkan Gambar 4.20. dapat diketahui bahwa dengan menerapkan metode penanganan *stray node* dengan pertimbangan energi menghasilkan PDR yang lebih baik jika dibandingkan dengan tanpa penerapan metode tersebut. Namun dalam beberapa nilai *P* yang disimulasikan dengan dengan menerapkan metode penanganan *stray node* tanpa pertimbangan energi menghasilkan PDR yang lebih tinggi. Hal ini berkaitan dengan jumlah *cluster* baru yang dibentuk saat proses penanganan *stray node*.

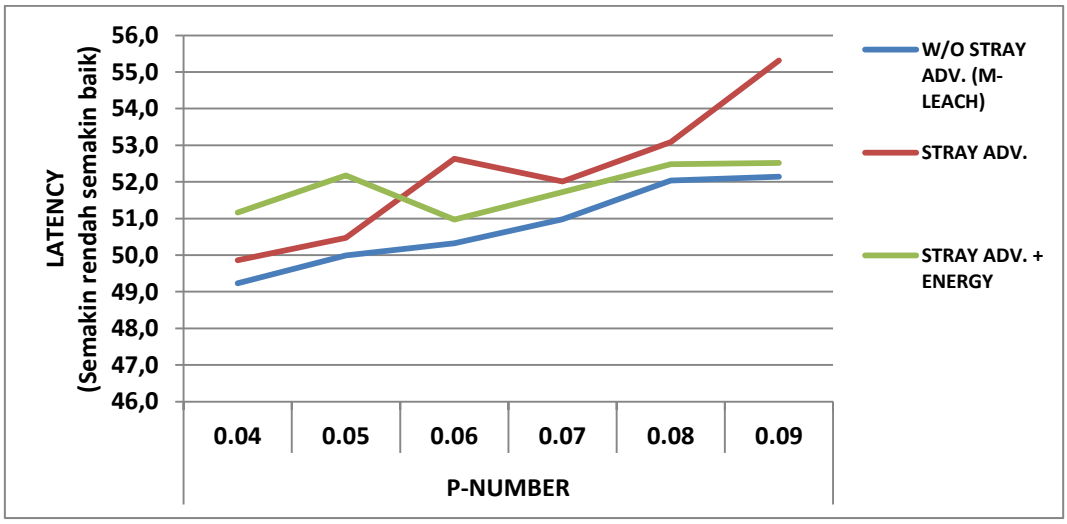
Tanpa mempertimbangkan faktor energi berarti jika ada beberapa *stray node* yang memiliki jumlah 15 tetangga yang sama-sama berstatus sebagai *stray node* maka seluruh node tersebut akan menjadi CH tanpa memandang sisi *residual energy*. Dengan banyaknya CH yang terbentuk maka akan hanya tersisa sedikit *stray node* pada jaringan dan paket yang dikirim menuju *sink* secara langsung akan lebih sedikit sehingga jaminan paket terkirim akan lebih tinggi. Namun demikian akan berimbas negatif pada konsumsi energi karena dengan

menjadi CH maka konsumsi energi individu *node* akan semakin tinggi seperti yang dijelaskan sebelumnya.



Gambar 4.20. PDR dari Hasil Pengujian dalam Nilai *P* Berbeda.

Dari Gambar 4.21 dapat diketahui bahwa dengan menerapkan metode penanganan *stray node* dengan pertimbangan energi maka *latency* jaringan lebih tinggi jika dibandingkan dengan tanpa menerapkan metode penanganan *stray node* teruatama pada nilai *P* yang digunakan rendah. Hal ini berhubungan dengan jumlah CH yang terbentuk dalam jaringan.



Gambar 4.21. *Latency* Hasil Pengujian dalam Nilai *P* Berbeda.

Pada saat nilai *P* yang digunakan bernilai rendah maka kemungkinan CH terbentuk pada beberapa *round* awal juga akan rendah sehingga akan lebih banyak *node* yang melakukan pengiriman secara langsung menggunakan *Distance-based* GPSR sehingga *latency* menjadi rendah. Sebaliknya pada nilai *P* tinggi maka kemungkinan terbentuk CH pada tiap *round* semakin tinggi sehingga lebih banyak CH dan lebih banyak paket akan dikirimkan melalui CH dan berakibat *latency* yang semakin tinggi. Namun perbedaan *latency* yang terjadi cenderung rendah dan bervariatif untuk nilai *P* yang disimulasikan dengan rata-rata perbedaan hanya 2ms sehingga pada jaringan yang berfokus terhadap efisiensi energi hal ini dapat dikatakan tidak signifikan.

[Halaman ini sengaja dikosongkan]

BAB V
KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan yang dapat diambil berdasarkan pada penelitian yang telah dilakukan. Dalam Bab 5 ini diuraikan tentang hal-hal yang perlu dipertimbangkan untuk pengembangan penelitian lebih lanjut. Penjelasan yang lebih terperinci tentang hal-hal tersebut diuraikan pada sub-bab berikut.

5.1.Kesimpulan

Berdasarkan hasil ujicoba dan analisis yang telah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Masalah *multiple* CH diselesaikan secara *energy-aware* dengan cara membandingkan faktor *residual energy* dari CH, dimana CH dengan *residual energy* tertinggi akan dipilih oleh *member node*.
2. Masalah *stray node* dapat diselesaikan dengan mempertimbangkan faktor energi dimana *stray node* yang memiliki sejumlah tetangga *stray node* lainnya dengan tingkat energi lebh rendah akan menjalankan fase LEACH *setup* ulang. Selain itu pada saat proses komparasi nilai *random* [0-1] dengan LEACH *threshold* dalam fase LEACH *setup* ulang tersebut *residual energy* dari *stray node* calon CH akan dipertimbangkan sebagai pengurang nilai *random*, bertujuan agar jika tingkat energi suatu *stray node* semakin tinggi maka kemungkinan dirinya untuk menjadi CH bagi para *stray node* lainnya semakin tinggi.
3. Modifikasi proses pemilihan CH dalam masalah *multiple* CH dengan mempertimbangkan faktor energi dapat meningkatkan performa jaringan dalam hal berikut:
    a. *Network lifetime* ditandai dengan waktu *node* pertama mati dalam jaringan rata-rata meningkat 186 menit dibandingkan dengan penggunaan paramater *distance*.
    b. Jumlah rata-rata paket dikirim dan diterima mengalami peningkatan masing-masing 9,49% dan 12,19% lebih banyak.

c. *Packet delivery ratio* rata-rata mengalami peningkatan 2,167% jika dibandingkan dengan penggunaan faktor *distance* (82% vs 84,167%).

4. Implementasi metode penanganan *stray node* dengan pertimbangan energi dapat meningkatkan performa jaringan dalam hal berikut:
    a. *Network lifetime* ditandai dengan waktu *node* pertama mati dalam jaringan rata-rata meningkat 779 menit jika dibandingan dengan tanpa penerapan metode tersebut.
    b. *Residual energy* meningkat pada T-1000, T-2000, dan T-3000 masing-masing 1,7%, 2,4%, dan 1,9% jika dibandingkan dengan tanpa penerapan metode penanganan *stray node* dengan pertimbangan energi.
    c. *Packet delivery ratio* rata-rata mengalami peningkatan 3% jika dibandingkan dengan tanpa penerapan metode penanganan *stray node* dengan pertimbangan energi (80% vs 83%).

5.2. Saran

Dari penelitian yang sudah dilakukan, beberapa hal yang mungkin masih bisa dikembangkan adalah sebagai berikut:

- Pengaplikasian metode routing selain *Distance-based* GPSR yang lebih efisien.
- Mekanisme penanganan *hole* yang dapat terjadi jika *node* yang posisinya dekat dengan *sink* telah mati.

DAFTAR PUSTAKA

Abushiba, W., Johnson, P., Alharthi, S., & Wright, C. (2017). An EnergyEfficient and Adaptive Clustering for Wireless Sensor Network (CH-leach) using Leach Protocol. *Computer Engineering Conference (ICENCO) 13th International*.

Anasane, A. A., & Satao, R. A. (2016). A Survey on Various Multipath Routing Protocols in Wireless Sensor Networks. *Procedia Computer Science, Vol.79*, 610-615.

Dietrich, I., & Dressler, F. (2009). On the Lifetime of Wireless Sensor Networks. *ACM Transactions on Sensor Networks Vol.5 No.1*, 1-38.

Ghica, O. C. (2010). *SIDnet-SWANS Manual.* SIDnet-SWANS Manual.

Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-Efficient Communication Protocol for Wireless Microsensor Networks. *Proceedings of the Hawaii International Conference on System Sciences*.

Jino Ramson, S., & Jackuline Moni, D. (2017). Applications of Wireless Sensor Networks – A Survey. *IEEE International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology ICIEEIMT*.

Karp, B. (2000). GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. *Proceedings of Sixth Annual International Conference on Mobile Computing & Networking, Boston*, 243-254.

Karray, F. J.-O. (2018). A comprehensive survey on wireless sensor node hardware platform. *Computer Networks, Vol.144*, 89–110.

Liu, X. (2012). A Survey on Clustering Routing Protocols in Wireless Sensor Networks. *Sensors 2012*.

Maimour, M. (2018). Interference-aware multipath routing for WSNs: Overview and performance evaluation. *Applied Computing and Informatics*.

Noor-ul-Sabah, T. A. (2018). Energy Efficient Multiple Cluster Head Selection Routing Protocol. *International Journal of Multidisciplinary Sciences and Engineering. Vol. 9, No. 1*.

Palan, N. G., Barbadekar, B. V., & Patil, S. (2017). Low Energy Adaptive Clustering Hierarchy (LEACH) Protocol: A Retrospective Analysis. *International Conference on Inventive Systems and Control (ICISC)*.

Raed, M. B., & Abdalraheem, A. I. (2013). A Survey On LEACH-Based Energy Aware Protocols for WIreless Sensor Networks. *Journal of Communication Vol. 8, No. 3*.

Rajagopalan, R., & Varshney, P. K. (2006). Data Aggregation Techniques in Wireless Sensor Network: A Survey. *IEEE COmmunications Surveys and Tutorials*, 48-63.

Rault, T., Bouabdallah, A., & Challal, Y. (2014). Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks Vol.67*, 104-122.

Shukla, K. (2013). Research On Energy Efficient Routing Protocol LEACH For Wireless Sensor Networks. *IJERT*.

Tian, D. G. (2003). Energy efficient routing with guaranteed delivery in wireless sensor networks. *IEEE Wireless Communications and Networking*.

Wibisono, W., Ahmad, T., & Anggoro, R. (2018). Position-Based Scheme for Multi-Hop Routing Protocol in CLuster-Based Wireless Sensor Networks. *4th International Conference on Wireless and Telematics (ICWT)*.

BIOGRAFI PENULIS

Ryan Rizki Adhisa, adalah anak kedua dari dua bersaudara yang lahir di Tulungagung, Jawa Timur pada tanggal 3 April 1993. Penulis menempuh pendidikan dasar, pendidikan menengah pertama dan menengah atas di kota Pacitan, Jawa Timur, selanjutnya meneruskan pendidikan jenjang sarjana di Universitas Brawijaya di Kota Malang, Jawa Timur dengan jurusan Teknik Informatika, keminatan Sistem komputer. Setelah menyelesaikan pendidikan sarjana penulis melajutkan Pendidikan Magister (S2) di Institut Teknologi Sepuluh Nopember jurusan Teknik Informatika pada tahun 2016. Penulis memiliki minat di bidang Komputasi Berbasis Jaringan (KBJ). Penulis dapat dihubungi melalui ryan.adhisa@gmail.com. Instagram: @ryan.adhisa.

Sebagian dari tesis ini telah dipublikasikan dalam *paper* berjudul *"Energy Aware Multiple Cluster Head Selection and Stray Nodes Handling for LEACH Protocol in Wireless Sensor Network Environments"* pada IEEE *International Conference On Information Technology System And Innovation* (ICITSI) 2018 yang berlangsung di Kota Bandung pada tanggal 22 Oktober 2018.

[Halaman ini sengaja dikosongkan]

Lampiran 1. Sertifikat IEEE ICITSI 2018 untuk publikasi *paper "Energy Aware Multiple Cluster Head Selection and Stray Nodes Handling for LEACH Protocol in Wireless Sensor Network Environments"*.